Instituto Superior Técnico

Sistemas de Controlo de Tráfego
2018-2019

# 3º Relatório

# Posicionamento do recetor de GPS

Alunos:

Filipa Ribeiro, 84540

Corpo docente:

Fernando Duarte Nunes

# Índice

# 1 Introdução

O Sistema Global de Posicionamento, comummente designado por GPS, consiste num conjunto de 24 satélites agrupados em 6 órbitas diferentes de 4 satélites cada, cujos períodos orbitais são de 11 horas e 58 minutos. Assim sendo, garante-se que cada recetor de GPS tem, em cada instante, à sua disposição pelo menos 4 satélites, dos quais 3 estão encarregues pela triangulação da posição e o quarto satélite é responsável pela correção do relógio.

O recetor de GPS é um sistema passivo, isto é, não transmite qualquer sinal. O seu principal objetivo é estimar a sua posição e velocidade com elevado grau de acurácia, a partir de medições com ruído (pseudo-distâncias). No sistema de GPS, uma parte considerável dos erros advém das camadas atmosféricas ionosfera e troposfera, bem como da instabilidade dos relógios dos satélites em órbita.

Neste projeto será elaborado um programa em linguagem Matlab, com o intuito de se simular o funcionamento de um recetor GPS. Este recetor irá efetuar uma trajetória pré-definida e serão estimadas, em cada instante, a sua posição e velocidade, através de dois métodos iterativos diferentes: Mínimos Quadrados e Filtro de Kalman Linearizado, mais especificamente, modelo Posição-Velocidade (PV).

A performance destes dois métodos será avaliada em diversas situações diferentes. Nomeadamente, serão testados casos onde os erros ionosféricos estarão ou não presentes, bem como serão avaliados os efeitos que o o número de satélites e do valor do parâmetro Position Dilution of Precision (PDOP) têm na precisão das estimativas efetuadas.

# 2 Descrição teórica

## 2.1 Sistema de coordenadas

De forma a se resolverem as equações de navegação em recetores GPS, utilizaram-se, no total, 4 sistemas de coordenadas distintos: ECI (Earth Centered Inertial), ECEF (Earth-Centered, Earth-Fixed), ENU (East, North, Up) e Geodésico (Latitude, Longitude e Altitude).

As conversões entre os sistemas de coordenadas mencionados anteriormente realizaram-se com base no modelo físico da Terra adoptado pelo sistema GPS, isto é, o *world geodetic system* 1984 (WGS84), que aproxima a figura da Terra a uma elipsóide.

#### ECI - Earth Centered Inertial

O referencial ECI tem origem no centro de massa da Terra e o plano xy coincide com o plano equatorial da mesma. O eixo xx aponta fixamente para uma posição do céu, o equinócio vernal. Assim sendo, este referencial não apresenta movimento em relação às estrelas. Por sua vez, o eixo zz aponta para o pólo Norte e o eixo yy é determinado de forma a cumprir a Regra da Mão Direita. Este sistema de coordenadas é utilizado na determinação da posição dos satélites GPS nas suas orbitas.

#### ECEF - Earth Centered Inertial

O referencial ECEF tem origem no centro de massa da Terra contudo, ao contrário do referencial ECI, este acompanha o movimento rotacional da Terra. Neste referencial, o eixo xx aponta na direção cuja longitude é 0º e o eixo yy aponta na direção cuja longitude é 90º Este. À semelhança do referencial ECI, o eixo zz aponta também para o Pólo Norte. Este sistema de coordenadas é utilizado na resolução das equações de navegação.

#### ENU - East North Up

O sistema de coordenadas ENU é um referencial cujas coordenadas são expressas relativamente a um plano tangente à superfície terrestre, com origem num determinado ponto.

#### Geodésico

O referencial Geodésico é frequentemente utilizado para indicar pontos à superfície terrestre, pelo que as coordenadas do recetor de GPS surgem representadas desta mesma forma, isto é, latitude, longitude e altitude.

Assim sendo, no âmbito do cálculo da posição do recetor de GPS, foram essenciais as Toolbox's Aerospace e Mapping do programa Matlab de forma a serem implementadas as conversões: ECEF $\leftrightarrow$ Geodésico e ECEF $\leftrightarrow$ ENU. De seguida apresentam-se as conversões mencionadas anteriormente, as respetivas funções do programa Matlab utilizadas, bem como exemplos das suas aplicações neste projeto. Note-se que, a transformação ECI $\leftrightarrow$ ECEF será apresentada na secção (2.2).

- Geodésico → ECEF

  lla2ecef: Esta função recebe as coordenadas do recetor em Latitude, Longitude e Altitude e converte-as em coordenadas $X_{ecef}$, $Y_{ecef}$ e $Z_{ecef}$, de forma a serem utilizadas na obtenção da sub-constelação ótima, bem como nos algoritmos Mínimos Quadrados e Filtro de Kalman Linearizado.

- ECEF → Geodésico

  ecef2lla: Esta função recebe as coordenadas ECEF do recetor estimadas pelos algoritmos Mínimos Quadrados e Filtro de Kalman e converte-as em Latitude, Longitude e Altitude, de forma a serem utilizadas aquando da atualização do número de satélites em vista ao longo do percurso.

- ECEF → ENU

  ecef2enu: Esta função recebe as coordenadas do ponto de referência (recetor) em Latitude, Longitude e Altitude e as coordenadas de cada Satélite em ECEF e, por fim, converte estas últimas em coordenadas do sistema ENU, com o intuito de ser possível calcular os ângulos azimute e elevação de cada satélite em relação ao recetor.

- ENU → ECEF

  enu2ecef: Esta função recebe as coordenadas geodésicas do ponto de referência utilizado (ponto A), bem como recebe as coordenadas ENU de cada ponto do percurso AB-BC-CD-DE. De seguida, devolve em coordenadas ECEF de cada ponto da trajetória, de forma a serem utilizadas pelos algoritmos de estimação dos Mínimos Quadrados e do Filtro de Kalman Linearizado.

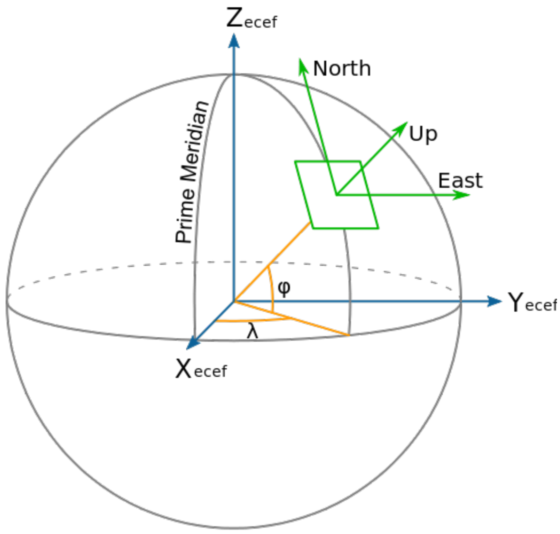



Figura 1: A azul, sistema de coordenadas ECEF. A amarelo, sistema de coordenadas Geodésico. A verde, sistema de coordenadas ENU [1].

## 2.2 Posição da constelação de satélites GPS

A informação relativa a cada satélite da constelação do Sistema de Posicionamento Global foi extraída de um Almanaque YUMA. É de se realçar o facto de que, só a partir do tempo de referência ($t_{oe}$) do Almanaque utilizado é que é possível determinar a posição dos satélites em órbita. Primeiramente, serão obtidas as posições aproximadas dos satélites em coordenadas do sistema ECI e, posteriormente, serão realizadas as conversões, também aproximadas, para o sistema de coordenadas ECEF.

De seguida, apresentam-se os parâmetros contidos nos Almanaques YUMA e utilizados na determinação das coordenadas dos satélites.

- $e_0$ : Excentricidade da órbita
- $\alpha$ : Inclinação da órbita
- $\dot{\Omega}$ : Taxa de mudança de longitude do nodo ascendente no tempo $t_{oe}$
- $A^{1/2}$ : Raíz quadrada do comprimento do semi-eixo maior da órbita
- $\Omega_0$ : Longitude do nodo ascendente no início da semana do GPS
- $w$ : Argumento do perigeu
- $M_0$ : Anomalia média no instante $t_{oe}$

O conjunto de equações seguintes permitem o cálculo aproximado das coordenadas dos satélites na sua órbita, no sistema ECI.

Em primeiro lugar, aquando do tempo de transmissão do sinal, $t_{st}$, a anomalia média M é dada por:

$$M = M_0 + n\Delta t \tag{1}$$

onde

$$\Delta t = t_{st} - t_{oe}$$

e $n$ é a velocidade de rotação da Terra que é constante e igual a:

$$n = \frac{2\pi}{T} = \sqrt{\frac{\mu}{A^3}} \tag{2}$$

Atribuiu-se à constante gravitacional da Terra, $\mu$, o seguinte valor:

$$\mu = 3.986005 \times 10^{14} m^3/s^2$$

A anomalia excêntrica $E$ pode ser obtida a partir de uma solução iterativa da equação de Kepler:

$$M = E - e_0 sin(E) \tag{3}$$

Este método iterativo segue a seguinte forma:

$$E_0 = M + \frac{e_0 sinM}{1 - sin(M + e_0) + sinM}$$

$$E_i = M + e_0 sin(E_{i-1}), \quad i = 1, 2, .., n$$

$$E = E_n \tag{4}$$

De seguida, uma vez que já é conhecido o valor da anomalia excêntrica $E$, é calculada a anomalia verdadeira $\nu$:

$$\nu = arctg_2\left(\frac{\sqrt{1 - e_0^2} sinE}{cosE - e_0}\right) \tag{5}$$

Por sua vez, a anomalia verdadeira $\nu$ é utilizada no cálculo do argumento da latitude, $\theta$. Este último, corresponde ao ângulo entre o nodo ascendente e a posição do satélite na sua órbita. Assim sendo, no tempo $t_{st}$ de transmissão de sinal tem-se:

$$\theta = \nu + w \tag{6}$$

Por outro lado, a longitude do nodo ascendente no tempo $t_{st}$ de transmissão de sinal é dado por:

$$\omega = \Omega_0 + \dot{\Omega}\Delta t - \dot{\Omega}_e t_{st} \tag{7}$$

O parâmetro $\Omega_e$ corresponde à taxa de rotação da Terra e, portanto, é igual a:

$$\Omega_e = 7.2921151467 \times 10^{-5} rad/s$$

Por último, o raio da órbita, isto é, a distância entre o satélite e o centro da Terra, é calculado pela fórmula:

$$R = A(1 - e_0 cosE) \tag{8}$$

Desta forma, conhece-se a longitude $\Omega$ do referencial ECI, bem como $R$, $\theta$ e $\alpha$ e, portanto, é possível determinar a posição de um satélite na sua respetiva órbita. Tal pode ser visualizado na figura (2).

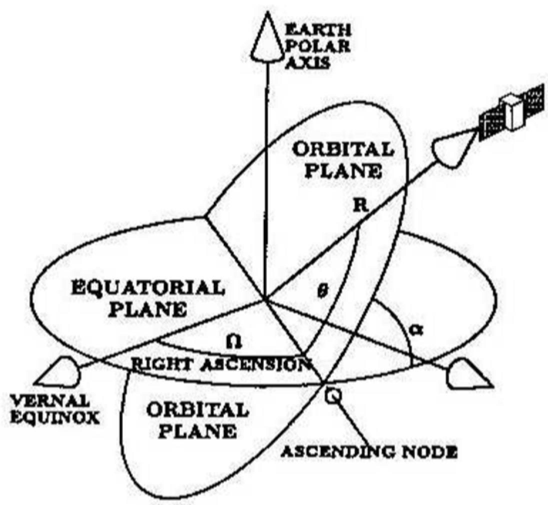



Figura 2: Coordenadas do satélite em função da longitude ECI, $\Omega$, e dos parâmetros $\theta$, $R$ e $\alpha$ [2].

Por fim, a posição aproximada de um satélite GPS em coordenadas ECEF é dada pela seguinte transformação:

$$\begin{bmatrix} X_{ecef} \\ Y_{ecef} \\ Z_{ecef} \end{bmatrix} = R \begin{bmatrix} \cos\theta\cos\Omega - \sin\theta\sin\Omega\cos\alpha \\ \cos\theta\sin\Omega + \sin\theta\sin\Omega\cos\alpha \\ \sin\theta\sin\alpha \end{bmatrix}$$

## 2.3 Determinação de satélites em vista

De forma a se determinar a sub-constelação de satélites em vista pelo recetor de GPS é necessário ter em conta o ângulo de máscara.

O ângulo de máscara consiste no ângulo entre o plano de horizonte do recetpor GPS e a direção recetor-satélite. Este ângulo indica um limite para a aceitação de mensagens provenientes dos satélites GPS em órbita. É importante atribuirem-se valores adequados ao ângulo de máscara, pois o sinal enviado por satélites que se encontrem abaixo do ângulo de máscara atravessa camadas mais espessas da atmosfera, como por exemplo, a ionosfera e a troposfera, que provocam alterações na velocidade da onda portadora do sinal, quando comparada com satélites em outras posições relativas [3].

A fim de ser possível comparar o ângulo de máscara com a elevação dos satélites, primeiramente será necessário calcular as coordenadas de cada satélite GPS num sistema referencial ENU, cujo ponto de referência é a posição do recetor, isto porque, o plano horizonte do recetor e o plano tangencial à sua posição são coincidentes.

O ângulo de elevação do satélite em relação ao horizonte do recetor (plano tangencial à posição do recetor) é dado por:

$$\varepsilon = arcsen\left(\frac{u}{\sqrt{u^2+n^2+e^2}}\right) \tag{9}$$

Por sua vez, o ângulo azimute do satélite é dado por:

$$\gamma = arctg_2\left(\frac{e}{n}\right) \tag{10}$$

Os ângulos azimute e elevação encontram-se representados na figura (3).

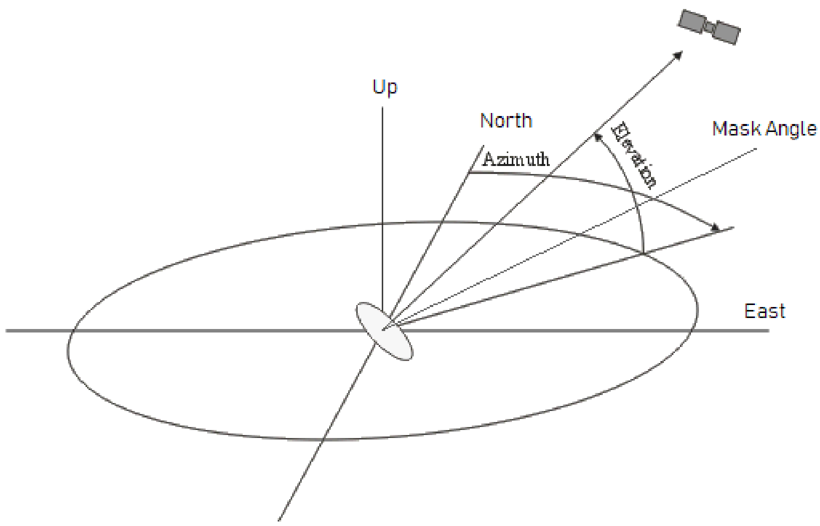



Figura 3: Representação dos ângulos azimute e elevação do satélite em relação ao recetor, bem como representação do ângulo de máscara a comparar. Sistema de coordenadas ENU.

## 2.4 Cálculo da sub-constelação ótima

A determinação de uma sub-constelação ótima dentro da constelação de satélites em vista é um processo importante no sitema GPS, pois, quando escolhida a melhor combinação de satélites a ser utilizada na estimação da posição do recetor, espera-se uma diminuição dos erros de posição do mesmo.

A obtenção da sub-constelação ótima em sistemas GPS é feita através de um algoritmo baseado no parâmetro Geometric Dilution Of Precision (GDOP). Este parâmetro apenas depende das relações geométricas entre os satélites e o recetor e é usado para caracterizar a precisão da estimativa da posição do recetor. Mais especificamente, este parâmetro traduz as relações dos erros de posição $\Delta X, \Delta Y$ e $\Delta Z$, bem como incorpora os erros de desvio do relógio do recetor em relação ao relógio do sistema. Assim sendo, quanto menor for o valor do parâmetro GDOP, menor serão os erros de posição e de desvio do relógio e, portanto, essa sub-constelação de satélites será a mais adequada para a estimação da posição do recetor.

Na figura (4) encontram-se esquematizados exemplos de possíveis geometrias satélites-recetor e os correspondentes valores qualitativos do parâmetro GDOP.

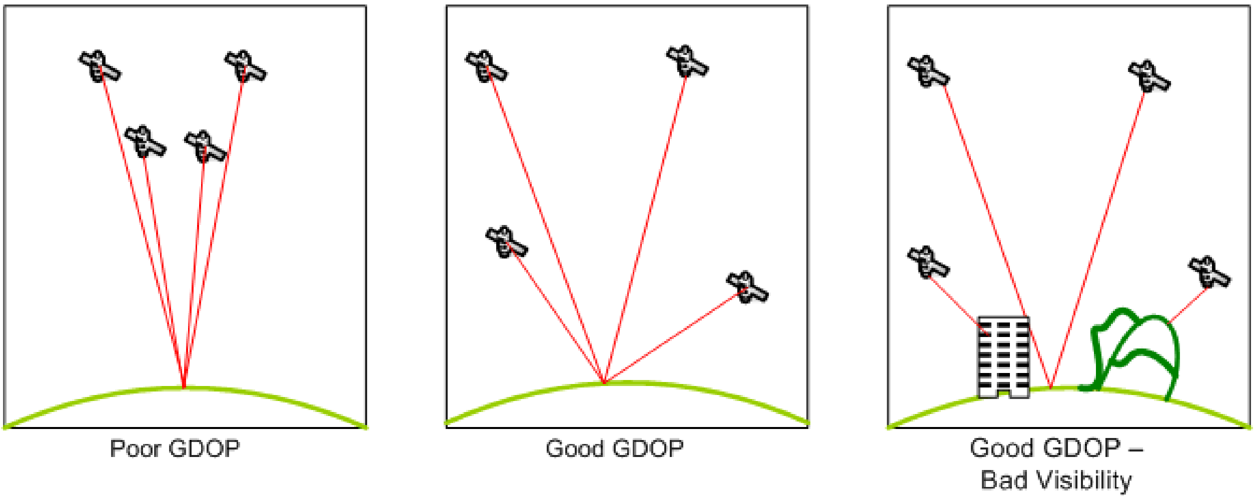



Figura 4: À esquerda, elevado (poor) GDOP. No centro, baixo (good) GDOP . À direita, baixo GDOP, mas baixa visibilidade. [4]

O parâmetro GDOP é o mais geral dos parâmetros Dilution of Precision (DOP). Neste trabalho, a determinação da sub-constelação ótima teve por base a minimização do parâmetro Position Dilution of Precision (PDOP). Este parâmetro é somente expresso em termos dos erros de posição segundo as coordenadas ($X_{ecef}$,$Y_{cef}$,$Z_{ecef}$), não incorporando, ao contrário do GDOP, os erros de desvios do relógio.

De seguida, apresentar-se-ão os métodos de cálculo dos parâmetros GDOP e PDOP, partindo do cálculo das pseudo-distâncias.

### Cálculo das pseudo-distâncias

De forma a determinar a posição do recetor ($x_u$, $y_u$, $z_u$) e o desvio do relógio do recetor em relação ao relógio do sistema $t_u$, são levadas a cabo medições das pseudo-distâncias entre o recetor e cada um dos N satélites, com N previamente selecionado, e que virão a fazer parte da sub-constelação ótima (N$\geq$4). Obtém-se o seguinte sistema de equações:

$$\rho_1 = \sqrt{(X_1 - x_u)^2 + (Y_1 - y_u)^2 + (Z_1 - z_u)^2} + ct_u + \epsilon_1$$

$$\rho_2 = \sqrt{(X_2 - x_u)^2 + (Y_2 - y_u)^2 + (Z_2 - z_u)^2} + ct_u + \epsilon_2$$

$$...$$

$$\rho_N = \sqrt{(X_N - x_u)^2 + (Y_N - y_u)^2 + (Z_N - z_u)^2} + ct_u + \epsilon_N \tag{11}$$

onde $\rho_i$ é a pseudo-distância, ($X_i$,$Y_i$,$Z_i$) é a posição do satélite $i$, com $i = 1, ..., N$ e $\epsilon_i$ é o erro associado à medição da correspondente pseudo-distância $\rho_i$. Note-se que aqui, quer as coordenadas do recetor, quer as coordenadas de cada satélite estão representadas no sistema de coordenadas ECEF.

Em geral, o erro $\epsilon_i$, associado à medição da pseudo-distância $\rho_i$ deve-se ao cancelamento incompleto dos erros introduzidos pelas camadas atmosféricas ionosfera e troposfera, dos erros do ruído térmico, entre outros. Neste trabalho, $\epsilon_i$, engloba os seguintes erros:

· $n_i$ : ruído Gaussiano, independente, de média nula e variância comum $\sigma^2_{noise}$

· $\rho_{iono,i}= \frac{10}{\sin \epsilon_i}$ $[m]$ : offset devido à ionosfera que depende da elevação do satélite $i$

Estas equações não lineares serão resolvidas por dois métodos diferentes: Mínimos Quadrados (2.5) e Filtro de Kalman linearizado (2.6).

### Cálculo dos parâmetros GPOP e PDOP

Partindo da equação (11), se desprezarmos os erros $\epsilon_i$ e se considarmos $b_u = ct_u$ obtém-se para os N satélites:

$$\rho_i = \sqrt{(X_i - x_u)^2 + (Y_i - y_u)^2 + (Z_i - z_u)^2} + b_u, \quad i = 1, ..., N \tag{12}$$

Linearizando-se a equação anterior através de uma expensão em série de Taylor e desprezando os termos não-lineares, obtém-se o seguinte modelo incremental:

$$\Delta\rho_i = \frac{\partial \rho_i}{\partial x_u}\Delta x_u + \frac{\partial \rho_i}{\partial y_u}\Delta y_u + \frac{\partial \rho_i}{\partial z_u}\Delta z_u + \frac{\partial \rho_i}{\partial b_u}\Delta b_u \tag{13}$$

e representando-o na sua forma matricial

$$\Delta\rho = G\Delta X \tag{14}$$

com

$$\Delta\rho = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ ... \\ \Delta\rho_N \end{bmatrix} \quad e \quad \Delta X = \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ \Delta b_u \end{bmatrix}$$

conclui-se que a matrix geométrica G é, então, representada da seguinte forma:

$$G = \begin{bmatrix} \frac{\partial \rho_1}{\partial x_u} & \frac{\partial \rho_1}{\partial y_u} & \frac{\partial \rho_1}{\partial z_u} & 1 \\ \frac{\partial \rho_2}{\partial x_u} & \frac{\partial \rho_2}{\partial y_u} & \frac{\partial \rho_2}{\partial z_u} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \rho_N}{\partial x_u} & \frac{\partial \rho_N}{\partial y_u} & \frac{\partial \rho_N}{\partial z_u} & 1 \end{bmatrix}, N \geq 4 \tag{15}$$

Uma vez que,

$$\frac{\partial \rho_i}{\partial x_u} = -\frac{X_i - x_u}{D_i}, \quad \frac{\partial \rho_i}{\partial y_u} = -\frac{Y_i - y_u}{D_i}, \quad \frac{\partial \rho_i}{\partial z_u} = -\frac{Z_i - z_u}{D_i}$$

e

$$D_i = \sqrt{(X_i - x_u)^2 + (Y_i - y_u)^2 + (Z_i - z_u)^2}$$

substituindo na equação (15) obtém-se:

$$G = \begin{bmatrix} -\frac{X_1 - x_u}{D_1} & -\frac{Y_1 - y_u}{D_1} & -\frac{Z_1 - z_u}{D_1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -\frac{X_N - x_u}{D_N} & -\frac{Y_N - y_u}{D_N} & -\frac{Z_N - z_u}{D_N} & 1 \end{bmatrix}, N \geq 4 \tag{16}$$

Considerando-se a matriz H igual a $(G^TG)^{-1}$:

$$H = (G^TG)^{-1} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix} \tag{17}$$

O valor do GDOP corresponde a:

$$GDOP = \sqrt{trace(G^TG)^{-1}} = \sqrt{h_{11} + h_{22} + h_{33} + h_{44}}$$

Por sua vez, o valor do PDOP corresponde a:

$$PDOP = \sqrt{h_{11} + h_{22} + h_{33}}$$

No âmbito deste projeto, de forma a se obter a sub-constelação ótima, deve-se calcular, para um conjunto de N satélites selecionados dentro dos $N_{View}$ satélites visíveis, todas as combinações de satélites possíveis. De seguida, para cada combinação de satélites obtida, calcula-se o seu PDOP e seleciona-se aquela que apresenta o menor valor.

## 2.5 Método dos mínimos quadrados

A solução da equação de posição (11) pode ser obtida de forma iterativa através do método dos mínimos quadrados. Neste projeto, o conjunto de N satélites tidos em conta na implementação deste método, foram os resultantes da sub-constelação ótima, calculadada na secção (2.4).

Partindo-se do demonstrado das equações (12) à (16) e relembrando-se, em particular, a equação (14) :

$$\Delta\rho = G\Delta X$$

verifica-se que a solução dos mínimos quadrados é dada por:

$$\Delta X = (G^TG)^{-1}G^T\Delta\rho \tag{18}$$

Para o sistema de equações (18) ser possível e determinado, o número de medições não pode ser inferior a 4. Através da equação (18), consegue-se estimar a posição do recetor de GPS ($\hat{x}$,$\hat{y}$,$\hat{z}$), bem como o atraso do relógio do recetor em relação ao relógio do sistema, $\hat{b}$, partindo de uma condição inicial ($\hat{x_0}$, $\hat{y_0}$, $\hat{z_0}$, $\hat{b_0}$).

De seguida, será explicado o algoritmo dos mínimos quadrados implementado:

1. Conhecimento da posição dos N satélites da sub-constelação ótima e cálculo do sistema de pseudo-distâncias (11):

$$\rho_{i,medido} = \sqrt{(X_i - x_u)^2 + (Y_i - y_u)^2 + (Z_i - z_u)^2} + b_u + \epsilon_i, \quad i = 1, ..., N$$

2. Fornecimento da posição inicial do recetor ($\hat{x}$, $\hat{y}$, $\hat{z}$, $\hat{b}$)=($\hat{x_0}$, $\hat{y_0}$, $\hat{z_0}$, $\hat{b_0}$)

3. Cálculo da matriz geométrica G aproximada:

$$\hat{G} = \begin{bmatrix} -\frac{X_1-\hat{x}}{D_1} & -\frac{Y_1-\hat{y}}{D_1} & -\frac{Z_1-\hat{z}}{D_1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -\frac{X_N-\hat{x}}{D_N} & -\frac{Y_N-\hat{y}}{D_N} & -\frac{Z_N-\hat{z}}{D_N} & 1 \end{bmatrix} \tag{19}$$

com

$$D_i = \sqrt{(X_i - \hat{x})^2 + (Y_i - \hat{y})^2 + (Z_i - \hat{z})^2}$$

4. Determinação da pseudo-distância estimada:

$$\rho_{i,estimado} = \sqrt{(X_i - \hat{x})^2 + (Y_i - \hat{y})^2 + (Z_i - \hat{z})^2} + \hat{b}, \quad i = 1, ..., N$$

5. Determinação do vetor erro das pseudo-distâncias:

$$\Delta\rho = \rho_{i,medido} - \rho_{i,estimado}$$

6. Determinação do vetor erro de posição:

$$\Delta X = (\hat{G}^T \hat{G})^{-1} \hat{G}^T \Delta\rho$$

7. Obtenção da posição estimada:

$$\begin{bmatrix} \hat{x_1} \\ \hat{y_1} \\ \hat{z_1} \\ \hat{b_1} \end{bmatrix} = \begin{bmatrix} \hat{x_0} \\ \hat{y_0} \\ \hat{z_0} \\ \hat{b_0} \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta b \end{bmatrix}$$

8. Substituição da condição inicial:

$$(\hat{x_0}, \hat{y_0}, \hat{z_0}, \hat{b_0}) = (\hat{x_1}, \hat{y_1}, \hat{z_1}, \hat{b_1})$$

9. Repetição do algoritmo (voltar ao ponto 1)

Na secção (3) será explicada a forma como este algoritmo foi implementado conjuntamente com o resto do programa elaborado.

## 2.6 Filtro de Kalman Linearizado

O Filtro de Kalman Linearizado é comummente utilizado como alternativa ao método dos mínimos quadrados, no que diz respeito à solução da equação de navegação (11). As medições ou observações são feitas em intervalos de tempo discretos (k). Neste projeto o intervalo de tempo discreto considerado foi de 1s. O Filtro de Kalman Linearizado, em cada iteração, realiza a expansão em série de Taylor das não-leariedades do modelo das observações, relativamente à mais recente estimativa do vetor de estado, retendo apenas os termos lineares. O vetor de estados varia consoante o modelo: Posição (P), Posição-Velocidade (PV) e Posição-Velocidade-Aceleração (PVA).

Neste projeto será implementado o modelo PV e, por conseguinte, o vetor de estados é constituído por 8 elementos, isto é, a posição do recetor $(x_{1,k}, x_{3,k}, x_{5,k}) = (x_u, y_u, z_u)$, a velocidade do recetor $(x_{2,k}, x_{4,k}, x_{6,k}) = (v_{ux}, v_{uy}, v_{uz})$ e os desvios do relógio do recetor em relação ao relógio do sistema $(x_{7,k}, x_{8,k}) = (b_u, \dot{b_u})$.

De seguida, serão apresentados os modelos de estado em tempo contínuo e discreto das coordenadas $(x_u, y_u, z_u)$ e do relógio do recetor, bem como o modelo de observação.

### Modelos de estado das coordenadas

Considere-se o modelo de estado em tempo contínuo para a coordenada $x_u$ do recetor:

$$\begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ u_v(t) \end{bmatrix}$$

A matriz de covariância do vetor de ruído é dada por:

$$E\{[0u_v]^T[0u_v]\} = \mathrm{Q}\delta(t-\tau) = q_v \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

O correspondente modelo de estado em tempo discreto é:

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} + \begin{bmatrix} u_{1,k} \\ u_{2,k} \end{bmatrix}$$

Por sua vez, a matriz de covariância do vetor de ruído da dinâmica é:

$$E\{[u_{1,k}u_{2,k}]^T[u_{1,k}u_{2,k}]\} = Q_k = q_v \Delta t \begin{bmatrix} \frac{(\Delta t)^2}{3} & \frac{\Delta t}{2} \\ \frac{\Delta t}{2} & 1 \end{bmatrix}$$

### Modelos de estado do relógio do recetor

O modelo do relógio do recetor é definido por um vetor de estados de dimensão 2 e o sua esquematização encontra-se representada de seguida.

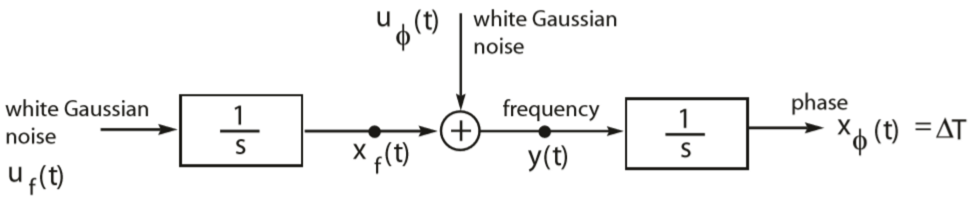



Figura 5: Modelo de estado do relógio do recetor de GPS.[5]

A equação da dinâmica é:

$$\begin{bmatrix} \dot{x_\phi}(t) \\ \dot{x_f}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_\phi \\ x_f \end{bmatrix} + \begin{bmatrix} u_\phi \\ u_f \end{bmatrix}$$

Os ruídos brancos $u_\phi$ e $u_f$ têm média nula, são independentes e caracterizados pela seguinte matriz de covariância:

$$Q_u = \begin{bmatrix} q_\phi & 0 \\ 0 & q_f \end{bmatrix}$$

A correspondente equação da dinâmica em tempo discreto é:

$$\begin{bmatrix} x_{\phi,k+1} \\ x_{f,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\phi,k} \\ x_{f,k} \end{bmatrix} + \begin{bmatrix} u_{\phi,k} \\ u_{f,k} \end{bmatrix}$$

E a covariância do ruído apresenta a seguinte forma:

$$\widetilde{Q}_k = \begin{bmatrix} q_\phi \Delta t + \frac{q_f(\Delta t)^3}{3} & \frac{q_f(\Delta t)^2}{2} \\ \frac{q_f(\Delta t)^2}{2} & q_f \Delta t \end{bmatrix}$$

com $q_\phi \approx \frac{h_0}{2}$ e $q_f = 2\pi^2 h_{-2}$.

Uma vez que, neste projeto, se utilizou o relógio do recetor com oscilador compensado em temperatura, então $h_0 = 2 \times 10^{-19}$ e $h_{-2} = 2 \times 10^{-20}$.

### Modelo de estado global

Assim sendo, extendendo o modelo de estado das coordenadas às restantes em falta $(y_u, z_u)$ e incorporando também o modelo de estado do relógio do recetor, obtém-se o modelo de estado completo para o modelo PV em tempo discreto:

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \\ x_{3,k+1} \\ x_{4,k+1} \\ x_{5,k+1} \\ x_{6,k+1} \\ x_{7,k+1} \\ x_{8,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & & & ... & & & 0 \\ & 1 & & & & & & \\ & & 1 & \Delta t & & & & \\ & & & 1 & & & & \\ \vdots & & & & 1 & \Delta t & & \vdots \\ & & & & & 1 & & \\ & & & & & & 1 & \Delta t \\ 0 & & & & ... & & & 1 \end{bmatrix} \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ x_{3,k} \\ x_{4,k} \\ x_{5,k} \\ x_{6,k} \\ x_{7,k} \\ x_{8,k} \end{bmatrix} + \begin{bmatrix} u_{1,k} \\ u_{2,k} \\ u_{3,k} \\ u_{4,k} \\ u_{5,k} \\ u_{6,k} \\ u_{7,k} \\ u_{8,k} \end{bmatrix} \tag{20}$$

E a matriz de covariância do vetor de ruído da dinâmica:

$$\mathrm{Q}_k = \begin{bmatrix} q_v\frac{(\Delta t)^3}{3} & q_v\frac{(\Delta t)^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ q_v\frac{(\Delta t)^2}{2} & q_v\Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & q_v\frac{(\Delta t)^3}{3} & q_v\frac{(\Delta t)^2}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & q_v\frac{(\Delta t)^2}{2} & q_v\Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q_v\frac{(\Delta t)^3}{3} & q_v\frac{(\Delta t)^2}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & q_v\frac{(\Delta t)^2}{2} & q_v\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \left[q_\phi\Delta t + \frac{q_f(\Delta t)^3}{3}\right]c^2 & \frac{q_f(\Delta t)^2}{2}c^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{q_f(\Delta t)^2}{2}c^2 & q_f\Delta t c^2 \end{bmatrix} \tag{21}$$

### Modelo de observação

Ao contrário das equações da dinâmica que são lineares independentemente do modelo da dinâmica adoptado, o modelo de observações é não linear e dado por

$$z_k = h[x(t_k)] + v_k$$

onde $z_k = [\rho_{1,k} \ ... \ \rho_{n,k}]^T$. Assim sendo, $z_k$ corresponde às pseudo-distâncias medidas no tempo $t = t_k$ e

$$h[x] = \begin{bmatrix} \sqrt{(X_i - x_a)^2 + (Y_i - x_b)^2 + (Z_i - x_c)^2} + x_d \\ \vdots \\ \sqrt{(X_n - x_a)^2 + (Y_n - x_b)^2 + (Z_n - x_c)^2} + x_d \end{bmatrix}$$

com $4 \leq n \leq \mathrm{N}$ e sendo N a dimensão da sub-constelação ótima de satélites. Por sua vez, $x_a$, $x_b$, $x_c$, $x_d$ são as componentes do vetor de estado correspondentes a $x_u$, $y_u$, $z_u$ e $c\Delta t$, respetivamente.

Uma vez que, se consideraram as variâncias dos erros das diferentes medições das pseudo-distâncias todos iguais a $\sigma_{UERE}^2$, o ruído de observação $v_k$ é caracterizado pela seguinte matriz de covariância no tempo $t = t_k$

$$E\{v_k v_k^T\} = R_k = \sigma_{UERE}^2 I$$

A matriz de observação do Filtro de Kalman Linearizado é dada por:

$$H_k = \left[ \frac{h_i[\hat{x}(k|k-1)]}{x_j} \right] \tag{22}$$

onde $i$ tem a dimensão do número de satélites utilizados ,ou seja, i = 1,...,N.
Por sua vez, j corresponde ao número de estados do modelo em uso. Neste caso, o modelo utilizado é o PV e, portanto, j = 1,...,8.

De acordo com o modelo PV, a matriz (22) é re-escrita da seguinte forma:

$$H_k = \begin{bmatrix} a_{x_1} & 0 & a_{y_1} & 0 & a_{z_1} & a_{b_1} & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{x_N} & 0 & a_{y_N} & 0 & a_{z_N} & a_{b_N} & 0 & \end{bmatrix} \tag{23}$$

com

$$a_{x_i} = \frac{X_i - \hat{x_u}}{\hat{r_i}}, \quad a_{y_i} = \frac{Y_i - \hat{y_u}}{\hat{r_i}}, \quad a_{z_i} = \frac{Z_i - \hat{z_u}}{\hat{r_i}}$$

onde

$$\hat{r_i} = \sqrt{(X_i - \hat{x_u})^2 + (Y_i - y_u)^2 + (Z_i - z_u)^2}$$

e

$$[\hat{x_u} \;\; \hat{y_u} \;\; \hat{z_u} \;\; \hat{b_u}]^T = [\hat{x_1}(k|k-1) \;\; \hat{x_3}(k|k-1) \;\; \hat{x_5}(k|k-1) \;\; \hat{x_7}(k|k-1)]^T.$$

De seguida, apresenta-se o conjunto de passos a ter em conta na implementação do Filtro de Kalman Linearizado:

1. Cálculo da matriz $R_0$

2. Computação da matriz de covariância do vetor de ruído da dinâmica $Q_0$

3. Fornecimento das condições iniciais

$$[\hat{x_1}(0|-1) \;\; 0 \;\; \hat{x_3}(0|-1) \;\; 0 \;\; \hat{x_5}(0|-1) \;\; 0 \;\; \hat{x_7}(0|-1) \;\; 0]^T \quad , \quad P(0|-1)$$

4. Cálculo da matriz de observação $H_0$

5. Cálculo do vetor de observação estimado $\hat{z_0} = h[x(0|-1)]$

6. Cálculo do vetor de observação medido (pseudo-distâncias) $z_0 = [\rho_{1,0} ... \rho_{n,0}]$

7. Computação do Ganho de Kalman

$$K_0 = P_0(0|-1)H_0^T[H_0P_0(0|-1)H_0^T + R_0]^{-1}$$

8. Estimação do vetor de estados

$$\hat{X}(0|0) = \hat{X}(0|-1) + K_0(z_0 - \hat{z_0})$$

9. Cálculo da matriz de covariância da filtragem

$$P(0|0) = [I - K_0H_0]P(0|-1)[I - K_0H_0]^T + K_0R_0K_0^T$$

10. Determinação da predição do vetor de estados

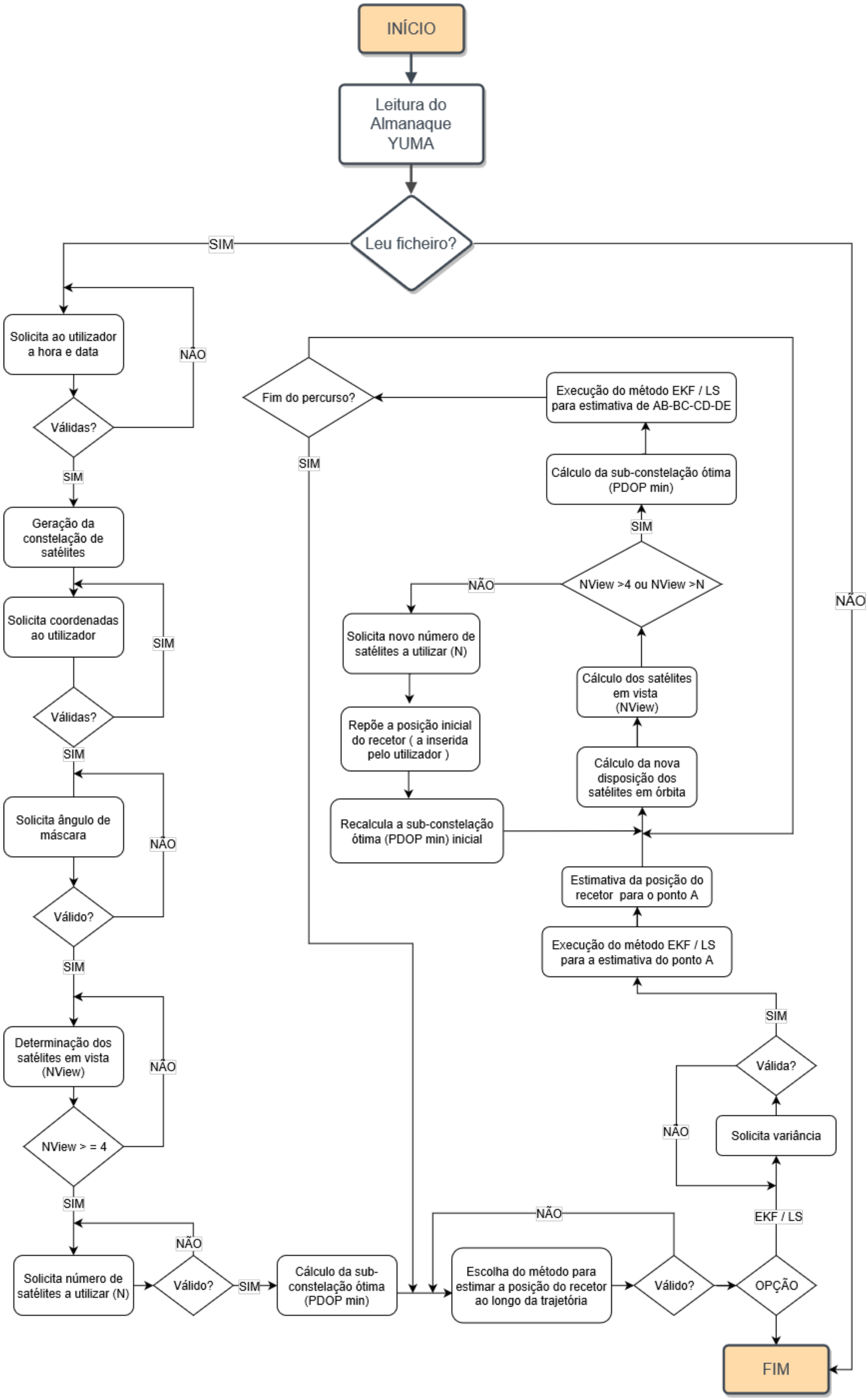
$$\hat{X}(1|0) = \Phi \hat{X}(0|0)$$

11. Cálculo da matriz de covariância do erro da predição

$$P(1|0) = \Phi P(0|0) \Phi^T + Q_k$$

De seguida, na secção (3), será explicada a forma como este algoritmo foi implementado conjuntamente com o resto do programa elaborado em linguagem Matlab.

# 3 Algoritmo implementado



# 4 Resultados obtidos

De seguida, serão apresentadas as disposições de todos os satélites GPS em órbita, bem como a constelação de satélites visíveis e sub-constelação ótima, para um determinado conjunto de valores de entrada. Este exemplo da execução do algoritmo da secção (3) tem por base o Almanaque YUMA do dia 13 de janeiro de 2019.

Uma vez implementados os algoritmos definidos nas secções (2.5), (2.6) e (3), serão comparadas as performances dos dois métodos de estimação utilizados, isto é, os Mínimos Quadrados e o Filtro de Kalman Linearizado. Será, também, analisado o impacto que o cenário de Canyon introduz na qualidade das estimativas, bem como, o efeito dos erros ionosféricos nas estimativas das posições e velocidades do recetor de GPS.

Conjunto de valores de entrada:

**Data**: 25-01-2019

**Hora**: 16:50:32

**Condição inicial da posição do recetor**: Instituto Superior Técnico de Lisboa

**Latitude**: 38º44'12 N

**Longitude**: 09º08'19 W

**Altitude**: 1500 m

**Ângulo de máscara**: 10º

## 4.1 Constelação de satélites GPS

Após leitura do Almanaque e recolha da data e hora de transmissão do sinal, foi possível o cálculo da posição dos satélites GPS em órbita, em coordenadas ECI. De seguida, foram calculadas as suas posições no sistema de coordenadas ECEF. Posteriormente, fez-se a conversão para coordenadas ENU, cuja referência é a posição do recetor, neste caso, o recetor encontra-se no Instituto Superior Técnico de Lisboa . Por último, foram calculados os ângulos de elevação e azimute de cada satélite em relação ao recetor. A sua representação gráfica encontra-se na figura (6).

## 4.2 Satélites em vista e sub-constelação ótima

De seguida, especificando o ângulo de máscara e comparando-o com a elevação de cada satélite, obtém-se o conjunto de satélites em vista. Para um ângulo de máscara igual a 10º, obtém-se um conjunto de 8 satélites visíveis ($N_{View} = 8$) representados a azul na figura (7).

Seguidamente, é escolhido o número $N$ de satélites GPS da sub-constelação ótima.

O critério de seleção da melhor combinação de $N$ satélites em $N_{View}$ satélites em vista, assenta na minimização do parâmetro PDOP. O valor de $N$ deve estar compreendido no seguinte intervalo de valores: $4 \leq N \leq N_{View}$.

A figura (8) demonstra a configuração de satélites ótima para um conjunto mínimo de 4 satélites, num total de 8 satélites em vista. O valor de PDOP mínimo correspondente é de 2.650. Quanto maior for o número $N$ de satélites escolhidos, menor será o PDOP mínimo, o que resultará em melhores estimativas da localização do recetor, pois os erros de posição são menores. O caso limite corresponde à escolha de $N = N_{View} = 8$. Para este conjunto de 8 satélites, o valor de PDOP mínimo é de 2.095. A sua representação é semelhante à da figura (8), diferindo apenas no facto de que todos os satélites (a azul) estariam selecionados (a vermelho).

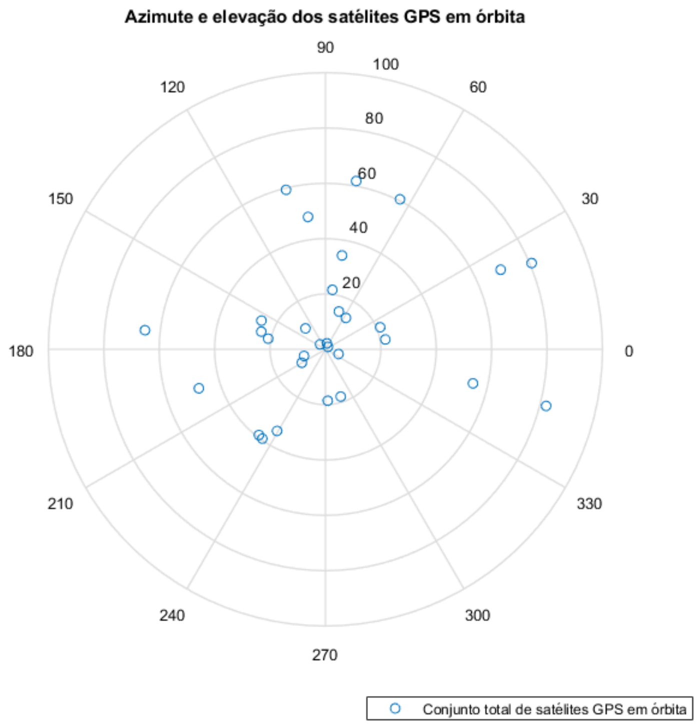



Figura 6: Representação polar da elevação (eixo radial) e azimute (eixo circular) dos satélites GPS em órbita.

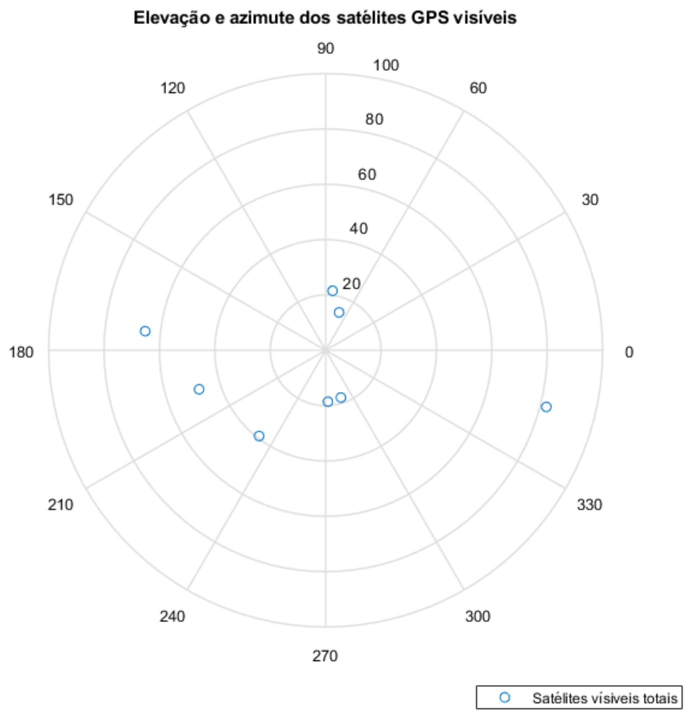



Figura 7: Representação polar da elevação (eixo radial) e azimute (eixo circular) dos satélites GPS visíveis, para ângulo de máscara igual a 10º.

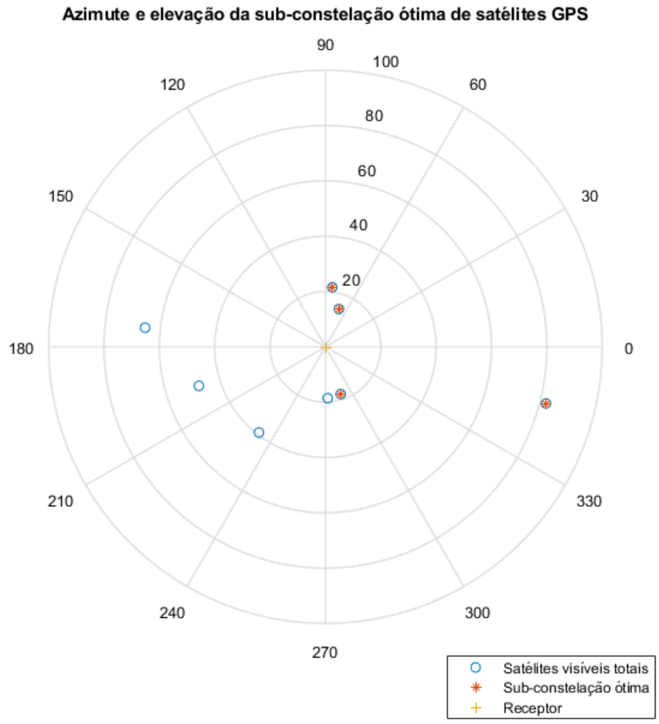



Figura 8: Representação polar da elevação (eixo radial) e azimute (eixo circular) da sub-constelação ótima de satélites, para N=4.

## 4.3 Percurso do recetor de GPS

O recetor de GPS realiza o seguinte percurso:

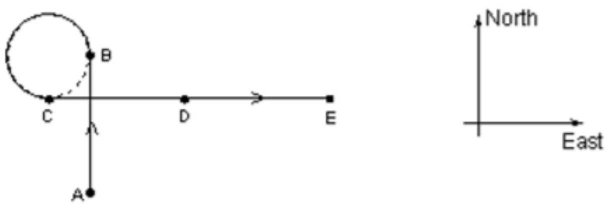



Figura 9: Trajetória AB-BC-CD-DE do recetor de GPS.

Com as devidas especificações:

- $\overline{AB}$: Movimento retilíneo uniformemente acelerado, com aceleração igual a 1 $ms^{-2}$ e velocidade inicial de 50 $ms^{-1}$. Duração: 100 segundos. Ângulo de máscara: 10º.
- $\overline{BC}$: Movimento circular uniforme. Duração: 50 segundos. Ângulo de máscara: 10º.
- $\overline{CD}$: Movimento retilíneo uniforme. Duração 50 segundos. Ângulo de máscara: 10º.
- $\overline{DE}$: Movimento retilíneo uniforme. Duração 50 segundos. Cenário Canyon: serão utilizados os 4 satélites com maior elevação em relação ao recetor de GPS.

O movimento realiza-se no plano horizontal do sistema de coordenadas ENU e o recetor inicia o seu percurso no ponto A, cujas coordenadas geodésicas são:

Latitude: 40º N

Longitude: 09º W

Altitude: 2000 m

## 4.4 Análise da performance

Para o conjunto de valores de entrada definidos no início do capítudo (4) e escolhendo o número $N$ de satélites da sub-constelação ótima máximo, isto é, $N = N_{View} = 8$, obtêm-se as seguintes estimativas de posição e velocidade do recetor de GPS, para ambos os métodos de estimação implementados. Note-se que, para a condição inicial de posição do recetor fornecida ao programa, isto é, as coordenadas do Instituto Superior Técnico de Lisboa, o recetor se encontra bastante distante do ponto A e, portanto, a partir daí será estudada a rapidez de convergência dos algorimtos de estimação implementados.

Valores de entrada adicionais:

**Variância**: $\sigma^2_{UERE} = 225m^2$

**Cenário Canyon**: Em vigor.

Como se pode observar nos gráficos da figura (10), o Filtro de Kalman Linearizado é o método que apresenta as melhores estimativas da posição do recetor de GPS ao longo de todo o percurso, comparativamente às obtidas pelo método dos Mínimos Quadrados.

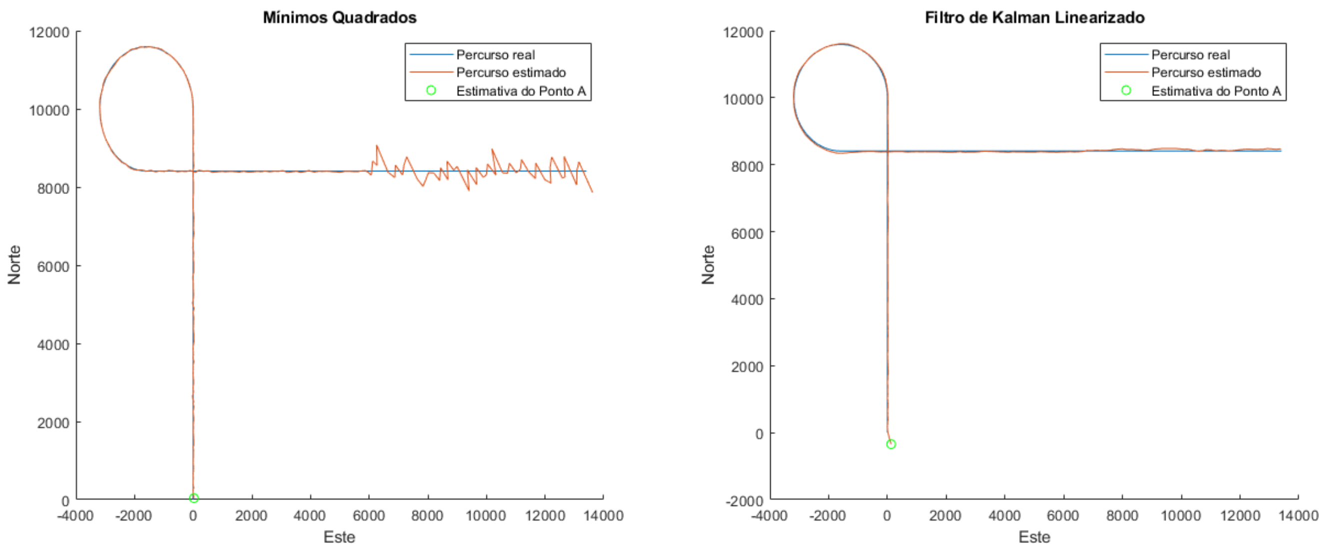



Figura 10: À esquerda, percurso real e estimativas do percurso e do Ponto A, segundo o método dos Mínimos Quadrados. À direita, percurso real e estimativas do percurso e do Ponto A, segundo o método do Filtro de Kalman Linearizado. Cenário Canyon no troço $\overline{DE}$.

Note-se que, nos resultados obtidos com o Filtro de Kalman Linearizado, a utilização dos 4 satélites em vista com maior elevação em relação ao recetor (cenário Canyon), praticamente não influência a qualidade da estimativa da posição do recetor de GPS. Em contrapartida, o método dos Mínimos Quadrados, para esse mesmo cenário, revela ser desastroso.

Nos troços $\overline{AB}$-$\overline{BC}$-$\overline{CD}$, o método dos Mínimos Quadrados estima de forma consistente a posição do recetor de GPS. Assim sendo, conclui-se que este método fornece, apenas, resultados aceitáveis quando utiliza uma sub-constelação ótima de satélites GPS no cálculo da solução da equação de navegação (11).

No que diz respeito à convergência dos algoritmos Mínimos Quadrados e Filtro de Kalman Linearizado, verifica-se, por análise dos gráficos da figura (11), que para uma condição inicial distante do Ponto A, ambos tendem rapidamente para o valor exato do ponto inicial da trajetória. Tal seria de se esperar, dado que, estes são métodos iterativos e a sua performance é tanto melhor quanto maior forem as distâncias recetor-satélite, pois, nesse caso, o processo de linearização presente quer no Filtro de Kalman Linearizado, quer nos Minimos Quadrados, comete erros menores. Assim sendo, em sistemas GPS, uma vez que as distâncias entre o recetor e o satélites de GPS são bastante elevadas, um pequeno número de iterações é suficiente para obter bons resultados. Neste caso, uma iteração resultou numa estimativa satisfatória do Ponto A.

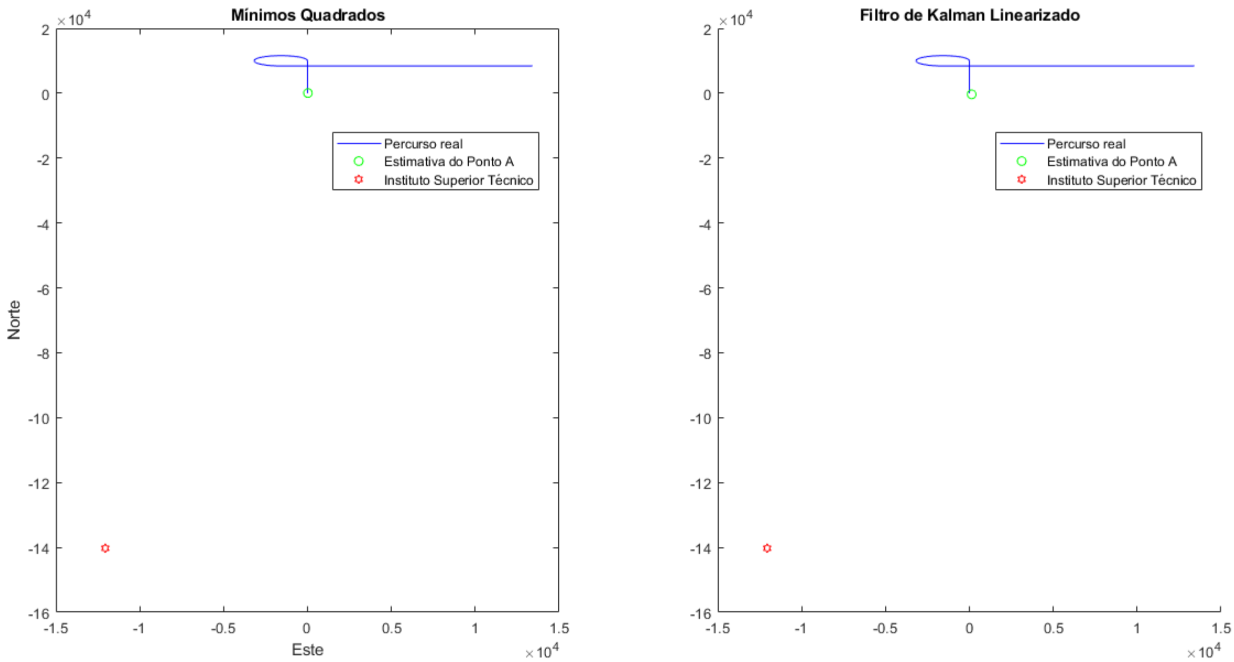



Figura 11: À esquerda, posição do recetor (Instituto Superior Técnico) e 1ª iteração da estimativa do Ponto A, segundo o método dos Mínimos Quadrados. À direita, posição do recetor (Instituto Superior Técnico) e 1ª iteração da estimativa do Ponto A, segundo o método do Filtro de Kalman Linearizado.

No método dos Mínimos Quadrados a estimativa da velocidade é feita por derivação parcial de posições adjacentes, isto é:

$$v(t) = \frac{X(t) - X(t-1)}{\Delta t}$$

Assim sendo, é previsível que os erros cometidos na sua estimação da velocidade sejam maiores do que os cometidos pelo método do Filtro de Kalman Linearizado, pois, para este último, as velocidades dos eixos , X, Y e Z, fazem parte do vetor de estados.

As estimativas da velocidade de ambos os métodos de estimação, encontram-se representadas nos gráficos da figura (12). Visto que, no troço $\overline{DE}$, o método dos Mínimos Quadrados parte de más estimativas da posição, então, era de se esperar que, para esse troço do percurso, a estimativa da velocidade, fosse também muito diferente da real (150 $ms^{-1}$).

Em contrapartida, o modelo PV do Filtro de Kalman Linearizado, estima de forma bastante satisfatória a velocidade do recetor de GPS.

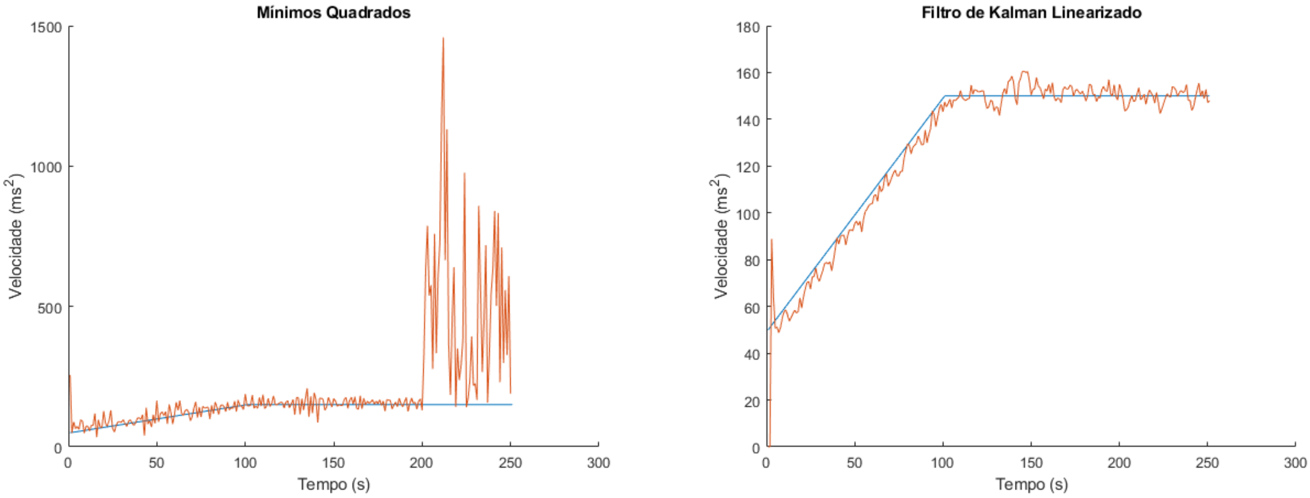



Figura 12: À esquerda, velocidade real e estimativa da velocidade, segundo o método dos Mínimos Quadrados. À direita, velocidade real e estimativa da velocidade, segundo o método do Filtro de Kalman Linearizado. Cenário Canyon no troço $\overline{DE}$.

Com o intuito de se avaliar a precisão dos modelos dos Mínimos Quadrados e do Filtro de Kalman Linearizado, foram calculadas as raízes quadradas dos erros quadráticos médios (RMSE) da posição e velocidade. Para o efeito, foram executadas 100 simulações e as fórmulas de cálculo utilizadas encontram-se representadas nas equações (24) e (25), com N igual ao número de pontos do percurso, isto é, 250 pontos. As suas representações gráficas serão apresentadas de seguida.

$$RMSE_{posico} = \sqrt{\sum_{n=1}^{N} \frac{[X(k) - \hat{x}(n)]^2 + [Y(k) - \hat{y}(n)]^2 + [Z(n) - \hat{z}(n)]^2}{N}} \tag{24}$$

$$RMSE_{velocidade} = \sqrt{\sum_{n=1}^{N} \frac{[V(n) - \hat{v}(n)]^2}{N}} \tag{25}$$

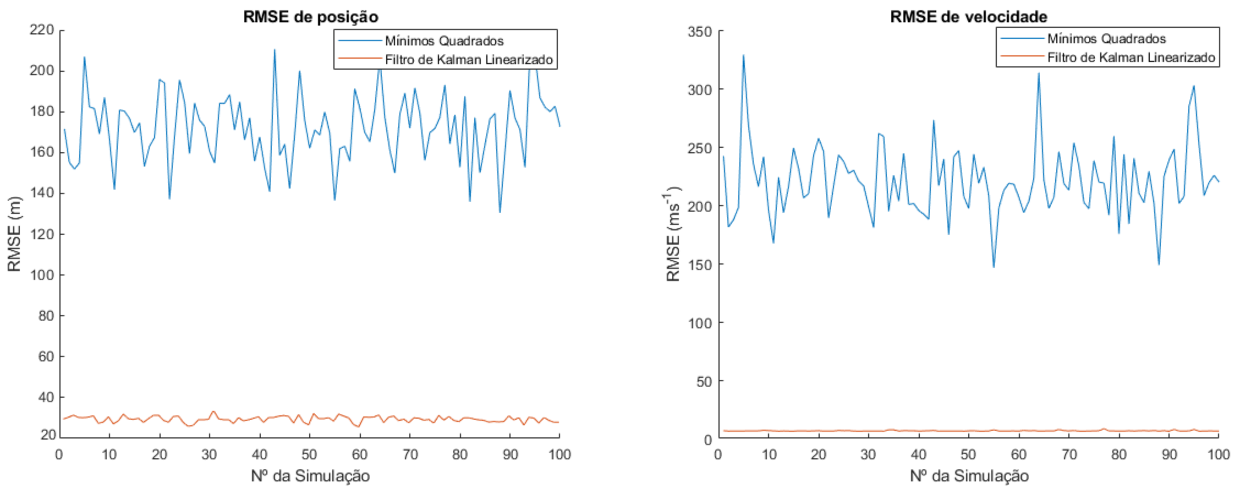



Figura 13: À esquerda, RMSE de posição em função do número da simulação, para o método dos Mínimos Quadrados e Filtro de Kalman Linearizado. À direita, RMSE de velocidade em função do número da simulação, para o método dos Mínimos Quadrados e Filtro de Kalman Linearizado.

Analisando-se os gráficos da figura (13), verifica-se uma grande discrepância nas performances dos dois métodos de estimação utilizados. A raiz dos erros quadrados médios de posição do método dos Mínimos Quadrados são mais do que 4 vezes superiores aos verificados pelo método do Filtro de Kalman Linearizado. Por sua vez, a raíz dos erros quadrados médios de velocidade do método dos Mínimos Quadrados, são cerca de 35 vezes superiores aos do método Filtro de Kalman Linearizado.

Assim sendo, sem margem para dúvidas, na eventualidade de um percurso com cenário Canyon, o método Filtro de Kalman Linearizado é o que dá garantias de estimativas adequadas de posição e velocidade do recetor de GPS.

## 4.5 Efeito do cenário Canyon

O ângulo de máscara limita o número de satélites GPS intervenientes na estimação da posição do recetor. Assim sendo, quanto maior for o ângulo de máscara, menor será o número de satélites em vista e, consequentemente, aqueles que se encontram em vista, apresentam ângulos de elevação mais elevados, face aos restantes satélites da constelação GPS. Desta forma, o que o cenário Canyon pretende introduzir neste trabalho é o estudo da influência que um ângulo de máscara elevado, e para o qual só é possível ter um número máximo de 4 satélites para a determinação da posição e velocidade do recetor, tem nas estimativas dos métodos dos Mínimos Quadrados e Filtro de Kalman Linearizado. Este cenário verifica-se na realidade, nomeadamente, em situações em que o recetor se encontre rodeado por prédios muito altos ou arranha-céus.

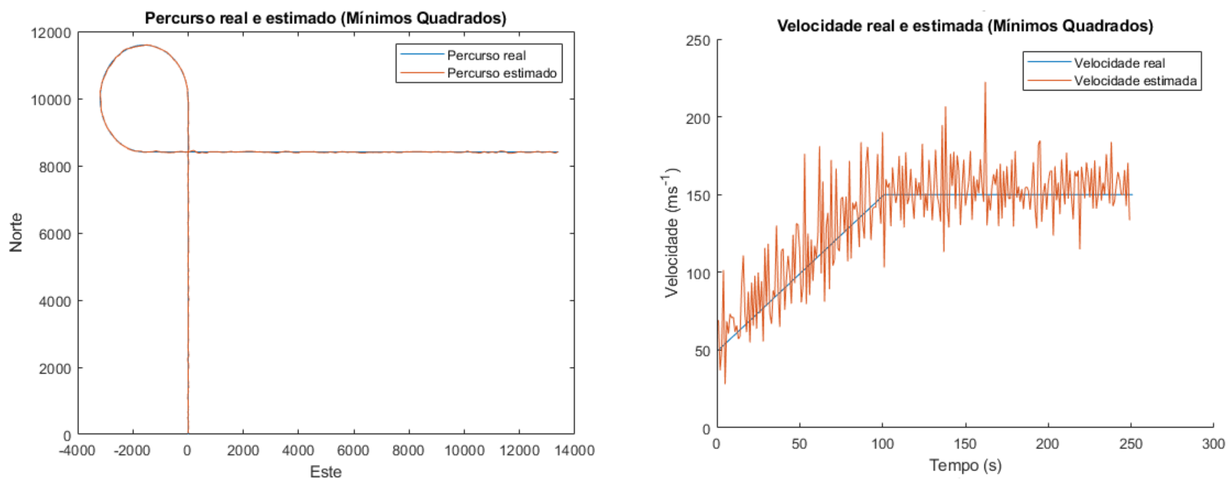



Figura 14: À esquerda, percurso real e estimado pelo método dos Mínimos Quadrados. À direita, velocidade real e estimada pelo método dos Mínimos Quadrados. Sem cenário Canyon no troço $\overline{DE}$.

Conclui-se que existe uma melhoria significativa da estimativa das posições e velocidades do troço $\overline{DE}$, por parte dos Mínimos Quadrados.

Como já fora mencionado anteriormente, através do gráfico à direita da figura (10), o efeito do cenário Canyon, praticamenete não afeta as estimativas do Filtro de Kalman Linearizado. Por essa razão, decidiu-se não apresentar os seus gráficos de estimação da posição e velocidade, uma vez que são semelhantes aos obtidos na secção anterior, figuras (10) e (12).

De seguida, apresentam-se os resultados RMSE de posição e velocidade apenas do troço $\overline{DE}$, **com** e **sem** cenário Canyon, para um conjunto de 100 simulações e para ambos os algoritmos de estimação.

- **RMSE de posição e velocidade do método Mínimos Quadrados no troço $\overline{DE}$**

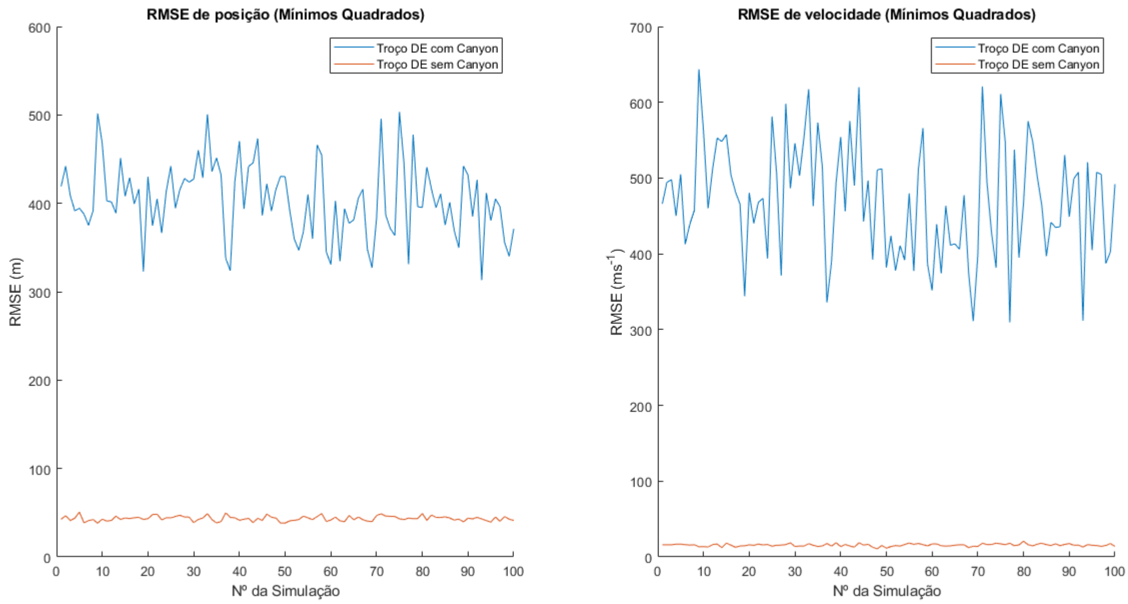



Figura 15: À esquerda, RMSE de posição do método dos Mínimos Quadrados, com e sem cenário Canyon. À direita, RMSE de velocidade do método dos Mínimos Quadrados, com e sem cenário Canyon.

Comprova-se assim, que em todo o conjunto das 100 simulações, existe uma melhoria abismal na qualidade de estimação quer da posição, quer da velocidade, no método dos Mínimos Quadrados, quando **não** são somente utilizados os 4 satélites de elevação máxima. No que diz respeito à posição, a raíz dos erros quadráticos médios desce de, aproximadamente, 400 metros para cerca de 50 metros, no troço $\overline{DE}$. Na velocidade, a raíz dos erros quadráticos médios desce de, aproximadamente, 450 $ms^{-1}$ para perto de 25 $ms^{-1}$, no troço $\overline{DE}$.

- **RMSE de posição e velocidade do método Filtro de Kalman Linearizado no troço $\overline{DE}$**

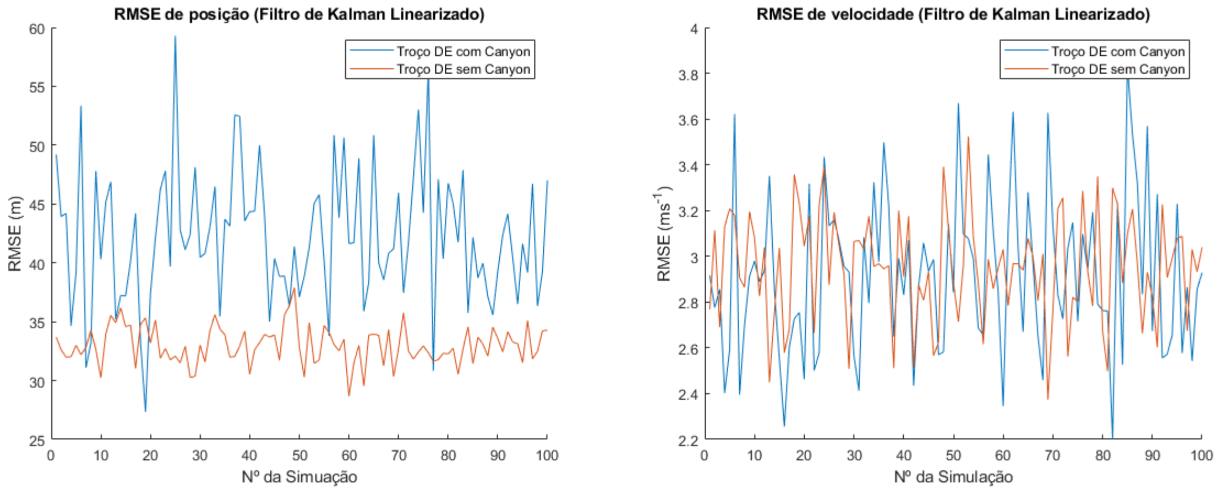



Figura 16: À esquerda, RMSE de posição do método Filtro de Kalman Linearizado, com e sem cenário Canyon. À direita, RMSE de velocidade do método Filtro de Kalman Linearizado, com e sem cenário Canyon.

Analisando-se os gráficos da figura (16), conclui-se que existem melhorias na estimativa da posição num cenário sem Canyon no troço $\overline{DE}$ no valor de aproximadamente 7 metros. Já nas estimativas da velocidade por parte do Filtro de Kalman Linearizado, não se verificam melhorias, sendo que os erros rondam, para os dois cenários possíveis, os 3 $ms^{-1}$.

## 4.6 Análise dos erros ionosféricos

Com o intuito de se estudar o efeito dos erros ionosféricos em ambos os algoritmos de estimação implementados neste projeto: Mínimos Quadrados e Filtro de Kalman Linearizado, foram realizadas 100 simulações com os dados de entrada previamente especificados, mas, agora, com a diferença da não contabilização dos erros desta camada atmosférica.

Relembre-se que, neste projeto, os erros da ionosfera foram modelados com recurso à seguinte equação:

$$\rho_{iono,m} = \frac{10}{sen(\varepsilon_m)} \tag{26}$$

onde m varia entre 1 e os N satélites utilizados e $\varepsilon_m$ representa o ângulo de elevação do satélite $m$ em relação ao recetor de GPS.

Os gráficos da figuras que se seguem, referem-se às 100 simulações **com** e **sem** contabilização de erros ionosféricos no percurso, para dois ângulos de máscara diferentes, e cenário Canyon no troço $\overline{DE}$, segundo os métodos dos Mínimos Quadrados e Filtro de Kalman Linearizado. São apresentados os valores RMSE das posições e velocidades ao longo de todo o percurso, em função do número da simulação.

- **RMSE de posição do método Mínimos Quadrados**

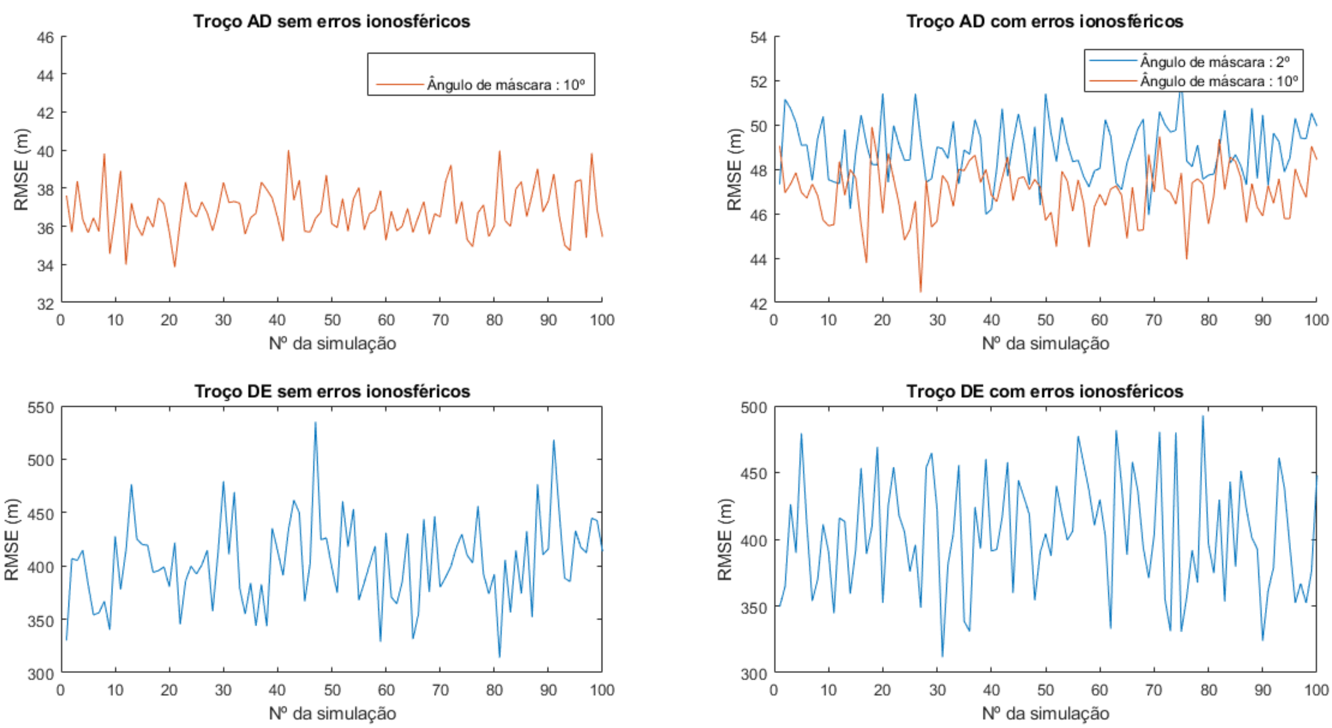



Figura 17: À esquerda, RMSE de posição sem contabilização de erros inosféricos para um ângulo de máscara de 10º. À direita, RMSE de posição com contabilização de erros inosféricos para ângulos de máscara de 2º e 10º. Modelo dos Mínimos Quadrados.

Como se pode verificar, através da análise do gráficos da figura (17), a contabilização dos erros introduzidos pela ionosfera, leva a um considerável aumento dos erros de posição ao longo de todo o percurso. Nos troços $\overline{AB}$, $\overline{BC}$, $\overline{CD}$, representados simplesmente por $\overline{AD}$, verifica-se, para um ângulo de máscara de 10º, um aumento da raíz do erro quadrático médio de posição em cerca de 10 metros, face ao cenário sem erros ionosféricos. No cenário Canyon do troço $\overline{DE}$, os resultados com e sem erros ionosféricos foram idênticos. Este resultado seria de se esperar, uma vez que, o cenário Canyon utiliza os 4 satélites com maior eleveção e, de acordo com a equação (26), quanto maior for a elevação dos satélites, menor é o erro introduzido pela ionosfera.

No que diz respeito à variação do ângulo de máscara dentro do troço $\overline{AD}$, para as condições onde existem erros ionosféricos, verifica-se que, para um ângulo de 10º, os erros são, em geral, 5 metros mais baixos que os erros para um ângulo de máscara de 2º. Este ângulo de máscara de 2º permite que satélites cuja elevação é muito baixa e que, portanto, os erros ionosféricos são elevados, participem no cálculo da posição do recetor, ao contrário do que acontece com o ângulo de máscara de 10º.

**- RMSE de velocidade do método Mínimos Quadrados**

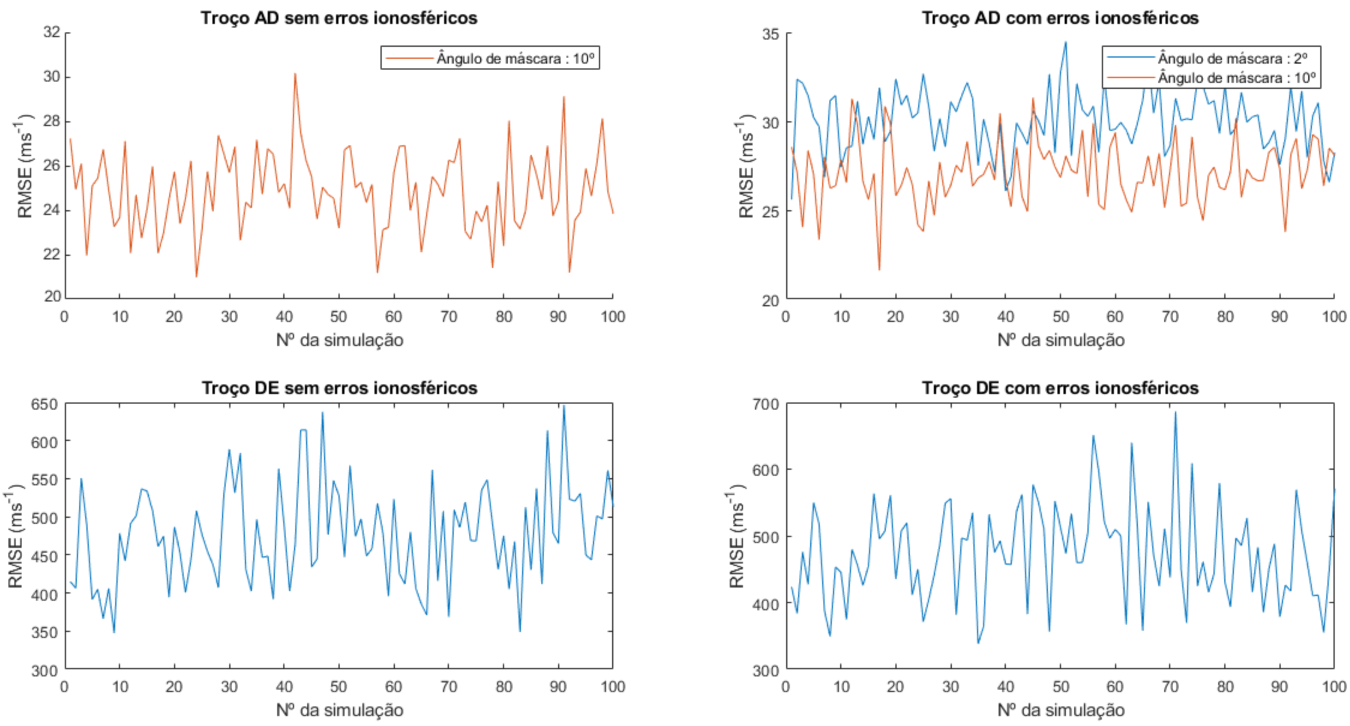



Figura 18: À esquerda, RMSE de velocidade sem contabilização de erros inosféricos para um ângulo de máscara de 10º. À direita, RMSE de velocidade com contabilização de erros inosféricos para os ângulos de máscara de 0º e 10º. Método dos Mínimos Quadrados.

Quanto à influencia da contibilização dos erros ionosférios na velocidade estimada pelo método dos Mínimos Quadrados, verifica-se que, no troço $\overline{AD}$, com e sem erros ionosféricos, a raíz do erros quadráticos médios diferia em cerca de 2 $ms^{-1}$. No cenário Canyon do troço $\overline{DE}$, os resultados do parâmetro RMSE são também idênticos, o que seria de se esperar, dado que a eleveção dos satélites, nesse cenário, é tão alta que os erros ionosféricos são relativamente baixos.

Nas condições com contabilização de erros ionosféricos no troço $\overline{AD}$, observa-se que, para uma diminuição do ângulo de máscara de 10º para 2º, os erros médios de velocidade aumentam. A razão pela qual isso ocorre é a mesma que a explicada anteriormente, em *RMSE de posição do método dos Mínimos Quadrados.*

- **RMSE de posição do método Filtro de Kalman Linearizado**

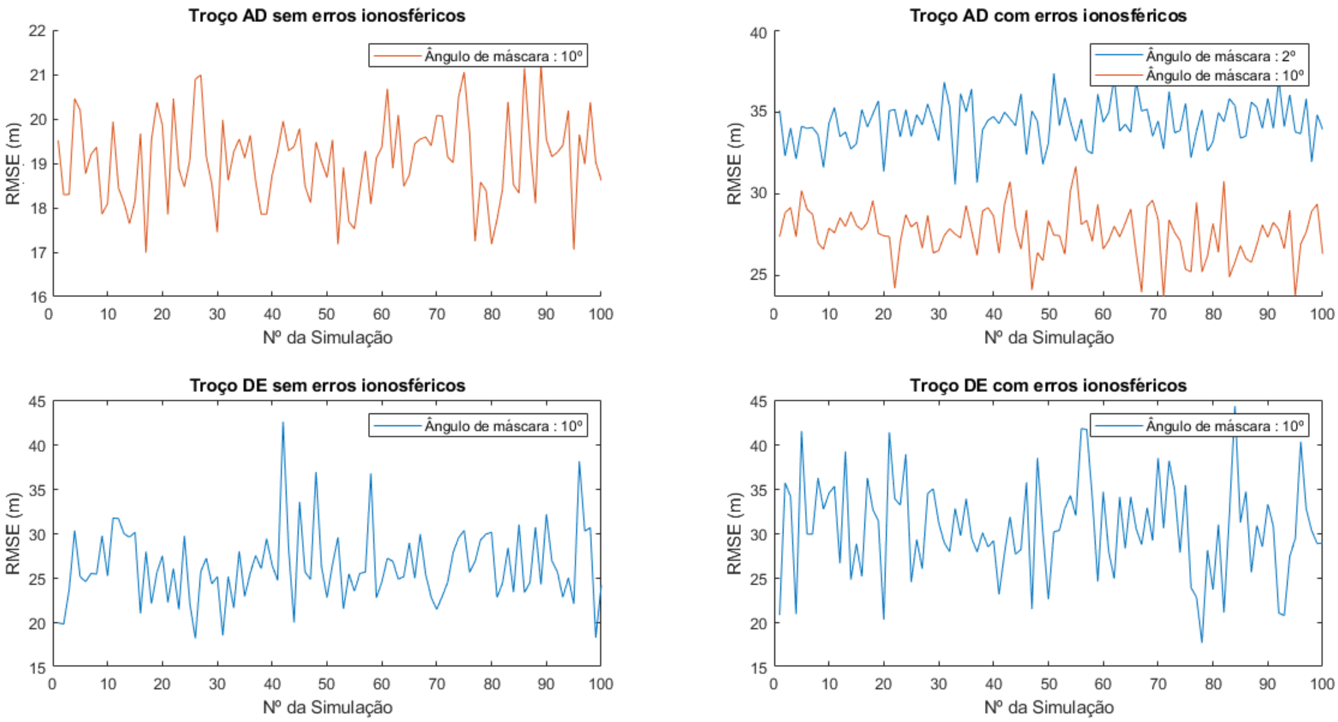



Figura 19: À esquerda, RMSE de posição sem contabilização de erros inosféricos para um ângulo de máscara de 10º. À direita, RMSE de posição com contabilização de erros inosféricos para ângulos de máscara de 2º e 10º. Método do Filtro de Kalman Linearizado.

Analisando-se os gráficos da figura (19), verifica-se que, para um ângulo de máscara de 10º, a raiz dos erros quadráticos médios de posição aumentam cerca de 10 metros.

Na comparação do troço $\overline{DE}$ com e sem erros inosféricos, verifica-se que os valores permaneceram praticamente inalterados, tal como seria esperado, devido ao cenário Canyon deste mesmo troço.

Nas condições com erros ionosféricos, no troço $\overline{AD}$, uma vez mais, a diminuição do ângulo de máscara levou a um aumento dos erros. Neste caso, verifica-se um aumento de aproximadamente 5 metros.

- **RMSE de velocidade do método Filtro de Kalman Linearizado**

Por último, por análise dos gráficos da figura (20), conclui-se que, para o troço $\overline{AD}$, existe um ligeiro aumento da raíz dos erros quadráticos médios de velocidade, de aproximadamente $0.5\ ms^{-1}$,

Novamente, o troço $\overline{DE}$ com cenário Canyon não apresenta diferenças significativas aquando da inclusão de erros ionosféricos. O mesmo se conclui, no troço $\overline{AD}$, no caso onde são incluídos os erros ionosféricos, em que a diminuição do ângulo de máscara de 10º para 2º, praticamente não influencia a raiz dos erros quadráticos médios das velocidades, verificando-se apenas alguns picos de erro esporádicos.

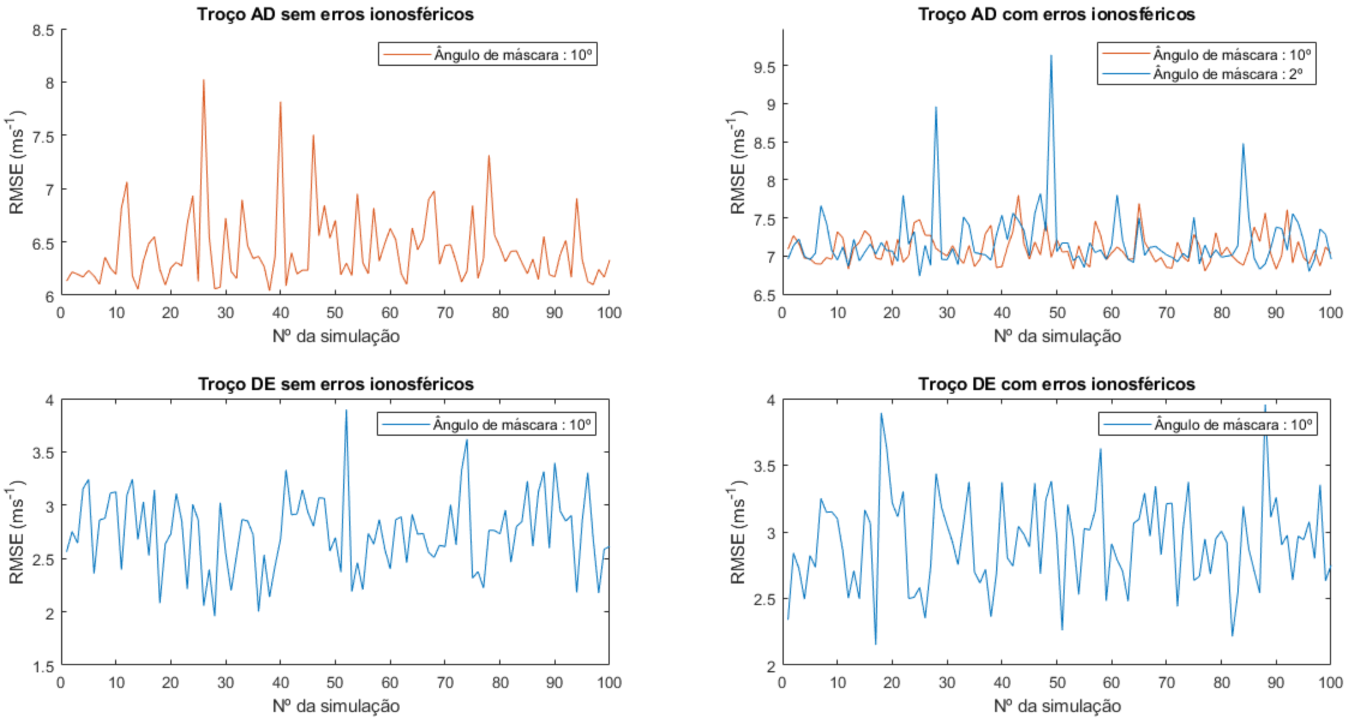



Figura 20: À esquerda, RMSE de velocidade sem contabilização de erros inosféricos para um ângulo de máscara de 10º. À direita, RMSE de velocidade com contabilização de erros inosféricos para ângulos de máscara de 2º e 10º. Método do Filtro de Kalman Linearizado.

# 5 Conclusão

Com o intuito de se revolver a equação de navegação em recetor de GPS, foram utilizados dois métodos iterativos diferentes: Mínimos Quadrados e Filtro de Kalman Linearizado (modelo PV).

O percurso do recetor de GPS dividia-se, essencialmente, em duas partes: o troço $\overline{AD}$, onde eram utilizados os satélites da sub-constelação ótima na estimativa das posições e velocidades do recetor, e o troço $\overline{DE}$, no qual o ângulo de máscara era de tal maneira elevado, que eram somente considerados os 4 satélites de GPS com maior ângulo de elevação nas estimativas efetuadas (cenário Canyon).

A performance dos dois algoritmos de estimação implementados neste trabalho, foi avaliada com base no parâmetro RMSE (Root-Mean-Square Error).

Verificaram-se, para as mesmas condições iniciais fornecidas ao programa, que o método do Filtro de Kalman Linearizado estimava tanto a posição como a velocidade do recetor de forma mais precisa. O facto de que no troço $\overline{DE}$ não era utilizada a sub-constelação ótima de satélites (geometria com menor PDOP) fazia com que o método dos Mínimos Quadrados deixasse de obter estimativas satisfatórias. Mais objetivamente, pode-se afirmar que em média, num percurso com o cenário Canyon, eram obtidas raízes dos erros quadráticos médios de posição no valor de aproximadamente 160 metros e raízes dos erros quadráticos médios da velocidade perto dos 200 $ms^{-1}$. Contudo, para um cenário onde, no percuso $\overline{DE}$, eram utilizados os satélites da sub-constelação ótima, as raízs dos erros quadráticos médios de posição baixavam para cerca de 50 metros e os de velocidade baixavam para valores médios entre 25 a 30 $ms^{-1}$.

Revela-se assim, a importância que o parâmetro PDOP, bem como a existência de um ângulo de máscara adequado, têm na qualidade das estimativas de posição e velocidade do recetor de GPS, quando é utilizado o método dos Mínimos Quadrados.

Por outro lado, o Filtro de Kalman Linearizado, não revelava ser afetado de forma significativa por este cenário Canyon, obtendo-se para a grande parte dos casos raízes dos erros quadráticos médios de posição na ordem dos 30 metros e de velocidade, geralmente entre, os 3 $ms^{-1}$ a 6 $ms^{-1}$.

O efeito dos erros ionosféricos foi também avaliado e verificou-se que este introduzia erros de posição de, aproximadamente, 10 metros e erros de velocidade. não muito significativos, por volta dos 2 $ms^{-1}$, no método dos Mínimos Quadrados. Para o método Filtro de Kalman Linearizado, as raízes dos erros quadráticos médios aumentavam cerca de 10 metros e 0.2 $ms^{-1}$ para a posição e velocidade, respetivamnente.

É de se salientar a importância da necessidade de modelos que representem adequadamente o comportamento da ionosfera, a fim de ser possível mitigar o seu impacto nas estimativas realizadas.

Atualmente, existem soluções que aumentam a acurácia das estimativas das posições e velocidades do recetor de GPS, que passam pelo uso de correções diferenciais nas medições básicas feitas aos sinais povenientes dos satélites. Este sistema designa-se por DGPS (differential GPS). É nesse sentido que a Engenharia deve evoluir.

# Referências


[1] Wikipédia. *Local tangent plane coordinates*. URL: https://en.wikipedia.org/wiki/Local_tangent_plane_coordinates#/media/File:ECEF_ENU_Longitude_Latitude_relationships.svg.

[2] Fernando Nunes. *V - GPS*. DECC, IST, 2018.

[3] *Topografia Civil*. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/3779573841777/Topografia_Civil_200910_S1_EP1_guia.pdf.

[4] *Basic GPS*. URL: https://nptel.ac.in/courses/105104100/lectureB_11/B_11_3GDOP.htm.

[5] Fernando Nunes. *I - Introduction to the Radio-Navigation Systems*. DECC, IST, 2018.

# 6 Código Matlab

## 6.1 Script principal

```
clear all;
clc;
header(); % Shows initial message

%% Reads YUMA Almanac in order to get information about satellites
data=read_almanac();

%% Request data and hour
ts0 = request_hour_data();

%% Satellite constellation creator:
% % Generation of a GPS satellite constellation for a given pre-defined
% % time with the use of almanacs obtained in Celestrak.com.
tsim = 0;
data = compute_orbit_parameters(data,ts0,tsim);
data = compute_ECEF_coordinates(data); % satellites ECEF coordinates
data0 = data;                          % saves data struct

%% Requests receiver's coordinates:
receiver_coord_lla = requests_coordinates_user(); % RECEIVER GEODETIC COORDINATES (LAT, LONG, ALT)

%% Determination of the satellites in view (NView) from a selected location (receiver_coord) and mask angle.
repeat = true;
while repeat
    mask_angle = requests_mask_angle();
    [NView, visible_sat, data] = satellites_in_view(receiver_coord_lla, data, mask_angle); % Sub-constellation
    fprintf('N mero de sat lites dispon veis: %d \n', NView);

    if NView<4
        fprintf ('O n mero de sat lites vis veis  insuficiente.\nTente um  gulo de m scara menor.\n\n')
    else
        repeat=false;
    end
end

%% Determination of the optimum sub-constellation for a certain number of satellits selectable by the user.
% % % Convert from receiver GEODETIC coordinates to receiver ECEF coordinates
ECEF_coord_r = lla2ecef(receiver_coord_lla,'WGS84'); % RECEIVER ECEF COORDINATES
```

```
ECEF_coord_r0 = ECEF_coord_r;                                    % SAVES INITIAL RECEIVER COORDINATES
ECEF_coord_r0(:,4)= 0;
%%% Determination of the optimum sub-constellation for a certain number of satellites N selectable (with NView >= N >=4) using minimization of the PDOP parameter.
%%%% Requests number N of satellits
N = requests_nr_sattelite( NView );
[minPDOP_sat_const, pdopmin ,total_comb,PDOP_vector] = PDOP_min(visible_sat, ECEF_coord_r, NView,N, 0 ); % Computes PDOP and chooses PDOPmin
minPDOP_sat_const0 = minPDOP_sat_const;                          % SAVES initial minPDOP_sat_const
creates_polar_graphs(visible_sat,minPDOP_sat_const,data);        % Shows polar graph with total visible satellites, satellites of optimum constellation and receiver position (origin)


%% TRAJECTORY SIMULATION

% FROM POSITION OF RECEIVER TO POINT A OF TRAJECTORY
pointA = lla2ecef([40, -9, 2000], 'WGS84');    % POINT A
pointA(:,4)=0;                                 % bu

path_enu = trajectory();                       % ENU path coordinates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               path_enu                                      %
% | xEast | yNorth | zUp | canyon_scenario | mask angle | velocity %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[path_ecef(:,1),path_ecef(:,2),path_ecef(:,3)] = enu2ecef(path_enu(:,1), path_enu(:,2), path_enu(:,3), 40, -9, 2000, wgs84Ellipsoid);
path_ecef(:,4:5) = path_enu(:,4:5);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               path_ecef                        %
%| xECEF | yECEF | zECEF | canyon_scenario | mask angle %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ECEF_coord_r(4)=0;                                       % bu

[ENU_RECEIVER(1),ENU_RECEIVER(2),ENU_RECEIVER(3)]= ecef2enu(ECEF_coord_r(1),ECEF_coord_r(2),ECEF_coord_r(3),40,-9,2000,wgs84Ellipsoid);

% plot(ENU_RECEIVER(1),ENU_RECEIVER(2), '+');

repeat_menu=true;
while repeat_menu

% ECEF_coord_r = ECEF_coord_r0;          % INITAL CONDITIONS RESTORED
```

```
EVERY TIME THE MENU IS PROMPT
% minPDOP_sat_const=minPDOP_sat_const0; % INITAL CONDITIONS RESTORED EVERY TIME THE MENU IS PROMPT
% data=data0;                          % INITAL CONDITIONS RESTORED EVERY TIME THE MENU IS PROMPT
% position_estimated=[];
clc;
option_modelo = menu();

                    % Path information:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      Mask Angle         %           Trajectory        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inserted by the user    %    from receiver position %
%     (mask_angle)        %           to point A        %
%$%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        10               %           AB-BC-CD          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Canyon Scenario     %              DE             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch option_modelo
    case '1' % LEAST SQUARES ALGORITHM
        variance = request_variance();

        %% Point A estimation
        ECEF_coord_r = least_square_algorithm( minPDOP_sat_const, variance, pointA , ECEF_coord_r );  % UPDATES RECEIVER COORDINATES
        LLA_coord_r(1,:)  = ecef2lla(ECEF_coord_r(1:3), 'WGS84');

        % Finally, the receiver arrives at point A.
        clc;
        position_estimated(1,:) = ECEF_coord_r;     % Final estimative of point A

        %% Trajectory ABDCE
        tsim = [0:251];
        disp('Chegou ao ponto A da trajetria.');
        disp('Clique numa tecla para continuar.');
        pause;
        disp('Comeando a trajetria  A-B-C-D-E ...');
        st=2;
        while st<length(tsim)
            data=compute_orbit_parameters(data,ts0,tsim(st));  % ts0 is the time inserted by the user, tsim is the time of the trajectory
            data=compute_ECEF_coordinates(data);                % satellites ECEF coordinates

```

```
            [NView, visible_sat] = satellites_in_view( LLA_coord_r(st-1,:), data, path_ecef(st,5)); % Sub-constellation

            if NView<4 || ( NView < N && ~isnan(path_ecef(st,5)))
                % if path_ecef(st,5) is a number then checks NView <4 or NView < N (AB-BC-CD trajectory)
                % if path_exef(st,5) is not a number (canyon scenario) then checks NView <4
                disp('Durante o percurso, o n mero de sat lites vis veis tornaram-se insuficientes, devido conjuga do gulo de m scara da trajet ria e n mero de sat lites inseridos.') % completar
                fprintf('Para o gulo de m scara pr -definido da trajet ria, insira um n mero de sat lites superior ou igual a 4 e inferior ou igual a %i. \n\n', NView);
                N = requests_nr_sattelite(NView);
                position_estimated = ECEF_coord_r; % Restarts estimation after point A
                st=2;                              % RESTARTS COUNTING
                minPDOP_sat_const  = PDOP_min( visible_sat, position_estimated(1,:), NView, N, path_ecef(2,5)); % Computes PDOP and chooses PDOPmin again
            else
                minPDOP_sat_const = PDOP_min( visible_sat, position_estimated(st-1,:), NView, N, path_ecef(st,5));
                position_estimated(st,:)= least_square_algorithm (minPDOP_sat_const, variance, path_ecef(st,:), position_estimated((st-1),:));
                LLA_coord_r(st,:)  = ecef2lla(position_estimated(st,(1:3)), 'WGS84');
                st=st+1;
            end
        end
        [ENU_position_estimated(:,1), ENU_position_estimated(:,2), ENU_position_estimated(:,3)] = ecef2enu (position_estimated(:,1),position_estimated(:,2), position_estimated(:,3), 40, -9, 2000, wgs84Ellipsoid);

        for i=1:250
        % v^2= vx^2+vy^2+vz^2
            velocity_ls(i,1)=sqrt((position_estimated(i+1,1)-position_estimated(i,1)).^2+(position_estimated(i+1,2)-position_estimated(i,2)).^2+(position_estimated(i+1,3)-position_estimated(i,3)).^2);
        end
        erro_ls(:,1)=sqrt((ENU_position_estimated(:,1)-path_enu(:,1)).^2+(ENU_position_estimated(:,2)-path_enu(:,2)).^2+(
```

```
        ENU_position_estimated(:,3)-path_enu(:,3)).^2);
    erro_velo_ls(:,1)=sqrt((path_enu(2:251,6)-velocity_ls(:,1)).^2);

  case '2'  % EKF (PV MODEL)
    variance = request_variance();
    clc;
    %% Point A estimation
    Pprev= 1*10^5*eye(8);
    % Dynamic matrix (phi)
    phi=eye(8);
    phi(1,2)=1;
    phi(3,4)=1;
    phi(5,6)=1;
    phi(7,8)=1;


    Xest_prev=zeros(8,1);
    Xest_prev(1,1) = ECEF_coord_r0(1);
    Xest_prev(3,1) = ECEF_coord_r0(2);
    Xest_prev(5,1) = ECEF_coord_r0(3);

    [position_est, Xest_prev, phi, Pprev] = kalman_filter(minPDOP_sat_const , variance , 0 , pointA , Xest_prev , phi, Pprev);
    position_est_lla = ecef2lla([position_est(1),position_est(3),position_est(5)],'WGS84');

    position_est_A = position_est;                                   % Saves estimation of point A

    % Finally, the receiver arrives at point A.
    clc;
    %% Trajectory ABDCE
    Pprev= 1*10^5*eye(8);
    % Dynamic matrix (phi)
    phi=eye(8);
    phi(1,2)=1;
    phi(3,4)=1;
    phi(5,6)=1;
    phi(7,8)=1;

    tsim = [0:251];
    disp('Chegou ao ponto A da trajetria.');
    disp('Clique numa tecla para continuar.');
    pause;
    disp('Comeando a trajetria  A-B-C-D-E ...');
    st=2;
    while st<length(tsim)                                            % Estimates point path(st) given postion_est(st-1)
```

```
            data=compute_orbit_parameters(data,ts0,tsim(st));  % ts0 is the time inserted by the user, tsim is the time of the trajectory
            data=compute_ECEF_coordinates(data);                % satellites ECEF coordinates

            [NView, visible_sat] = satellites_in_view(position_est_lla(st-1,:), data, path_ecef(st,5)); % Sub-constellation

            if NView<4 || ( NView < N && ~isnan(path_ecef(st,5)))
                % if path_ecef(st,5) is a number then checks NView <4 or NView < N (AB-BC-CD trajectory)
                % if path_exef(st,5) is not a number (canyon scenario) then checks NView <4
                disp('Durante o percurso o n mero de s a t lites v i s veis tornaram-se insuficientes')
                disp('devido   c o n j u g a  do  gulo  de  m scara da  t r a j e t ria  e n mero de  s a t lites inseridos.') % completar
                fprintf('Para o mesmo  gulo de m scara pre-definido na  t r a j e t ria, insira um n mero de  s a t lites superior ou igual a 4 e inferior ou igual a %i.\n\n', NView);
                N = requests_nr_sattelite(NView);
                st=2;
                Xest_prev= position_est_A;
                minPDOP_sat_const = PDOP_min(visible_sat, [position_est_A(1),position_est_A(3),position_est_A(5)], NView, N, path_ecef(st,5)); % Computes PDOP and chooses PDOPmin again
                Pprev= 1*10^5*eye(8);
                % Dynamic matrix (phi)
                phi=eye(8);
                phi(1,2)=1;
                phi(3,4)=1;
                phi(5,6)=1;
                phi(7,8)=1;
            else
                minPDOP_sat_const = PDOP_min(visible_sat, position_est(st-1,:), NView, N, path_ecef(st,5));
                [position_est(st,:), Xest_prev, phi, Pprev] = kalman_filter( minPDOP_sat_const , variance , 0 , path_ecef(st,:) , Xest_prev, phi, Pprev );
                position_est_lla(st,:)= ecef2lla([position_est(st,1),position_est(st,3),position_est(st,5)],'WGS84');
                st=st+1;
            end
        end
        [ENU_position_est(:, 1), ENU_position_est(:, 2),
```

```
            ENU_position_est(:, 3)] = ecef2enu(position_est(:,1),
            position_est(:,3), position_est(:,5), 40, -9, 2000,
            wgs84Ellipsoid);
        erro_ekf(:,1)=sqrt((ENU_position_est(:,1)-path_enu(:,1))
            .^2+(ENU_position_est(:,2)-path_enu(:,2)).^2+(
            ENU_position_est(:,3)-path_enu(:,3)).^2);
        velocity(:,1)=sqrt(position_est(:,2).^2+position_est(:,4)
            .^2+position_est(:,6).^2);
        erro_velo_ekf(:,1)=sqrt((path_enu(:,6)-velocity(:,1)).^2);

    case '3'
        clc;
        disp('* * * * * * * * * * * * * * * * * * * * * * * * * * * *
            * * * * * * * * * ');
        disp('* * * * * * * * * * * * * * * * * * GPS * * * * * * * *
            * * * * * * * * * ');
        disp('* * * * * * * * * * * * * * * * * * * * * * * * * * * *
            * * * * * * * * * ');
        disp('');
        disp('At  breve!');
        repeat_menu=false;


    %% PLOTS
%       %% Position
%       % LS
%       figure();
%       subplot(2,1,1);
%       hold all
%       plot(path_enu(:,1),path_enu(:,2),'b');
%       % position_estimated are ECEF coordinates
%       [ENU_position_estimated(:,1), ENU_position_estimated(:,2),
    ENU_position_estimated(:,3)] = ecef2enu (position_estimated(:,1),
    position_estimated(:,2), position_estimated(:,3), 40, -9, 2000,
    wgs84Ellipsoid);
%       plot(ENU_position_estimated(1,1), ENU_position_estimated(1,2)
    ,'o'); % Point A estimative
%       plot(ENU_position_estimated(:,1), ENU_position_estimated(:,2))
    ;
%       legend('Percurso real','Estimativa do ponto A', 'Estimativa do
     percurso');
%       xlabel('Este');
%       ylabel('Norte');
%       title('M nimos  Quadrados')
%       % EKF
%       subplot(2,2,1);
%       hold all
%       plot(path_enu(:,1),path_enu(:,2),'b');
%       % position_est is in ECEF coordinates
%       [ENU_position_est(:, 1), ENU_position_est(:, 2),
```

```
ENU_position_est(:, 3)] = ecef2enu(position_est(:,1),position_est(:,3), position_est(:,5), 40, -9, 2000, wgs84Ellipsoid);
%     plot(ENU_position_est(1,1),ENU_position_est(1,2),'o');  % Plots point A estimation
%     plot(ENU_position_est(:,1),ENU_position_est(:,2));      % Plots trajectory estimation
%     legend('Percurso real','Estimativa do ponto A', 'Estimativa do percurso');
%     xlabel('Este');
%     ylabel('Norte');
%     title('Filtro de Kalman Linearizado')

%     %% Velocity
%     %% LS
%     for i=1:250
%         % v^2= vx^2+vy^2+vz^2
%         velocity_ls(i,1)=sqrt((position_estimated(i+1,1)-position_estimated(i,1)).^2+(position_estimated(i+1,2)-position_estimated(i,2)).^2+(position_estimated(i+1,3)-position_estimated(i,3)).^2);
%     end
%     figure();
%     subplot(2,1,1);
%     hold all
%     plot(path_enu(:,6));
%     plot(velocity_ls);
%     legend('Velocidade real', 'Velocidade estimada');
%     xlabel('Tempo [s]');
%     ylabel('Velocidade (m/s)');
%     title('M nimos Quadrados')
%     %% EKF
%     velocity=sqrt(position_est(:,2).^2+position_est(:,4).^2+position_est(:,6).^2);  % v^2= vx^2+vy^2+vz^2
%     hold all
%     plot (path_enu(:,6));
%     plot(velocity);
%     legend('Velocidade real','Velocidade estimatada');
%     xlabel('Tempo (s)');
%     ylabel('Velocidade (m/s)');
%     title('Filtro de Kalman Linearizado')

end
end
```

## 6.2 Outras funções

```
function [data]=read_almanac

% Open ALMANAC document
f=fopen('almanac.txt');
if f == -1
    disp('Erro na abertura do ficheiro.');
end

while ~feof(f)
    tline=fgetl(f);
    str=strsplit(tline,':');
    if sum(str{1,1}=='*')>=1
        tline=fgetl(f);
        str=strsplit(tline,':');
        i=2;
        while i<14
            if strcmp(str{1,1},'ID')
                id=str2num(str{1,2});
                sat{id,1}=str2double(strtrim(str{1,2}));
            end
            tline=fgetl(f);
            str=strsplit(tline,':');
            sat{id,i}=str2double(strtrim(str{1,2}));
            i=i+1;
        end
    end
end
fclose(f);

%-----------------------------------------------------------------------%
% Extract data from sat cell array
var={'id','health', 'e', 'TOA', 'alpha', 'asc_r', 'sqrt_A', 'ASC', '
    w', 'M0', 'afo', 'af1', 'week'};
data=cell2struct(sat,var,2);
% Remove null rows
i=1;
while i<length(data)
    if isempty(data(i).id)
        data(i)=[];
        i=i-1;
    end
    i=i+1;
end
end
```

```
function time= request_hour_data
repeat =true;
while repeat
    data = input ('Insira a data [DD-MM-YYYY]: \n', 's');
```

```
    if length(data)==10
        if strcmp(data(3),'-')&& strcmp(data(6),'-')&& ~isnan(
            str2double(data(1:2)))&& ~isnan(str2double(data(4:5)))&&
            ~isnan(str2double(data(7:10)))
            repeat=false;
            day=str2double(data(1:2));
            month=str2double(data(4:5));
            year=str2double(data(7:end));
        else
            disp('Insira novamente.');
            repeat=true;
        end
    else
        disp('Insira novamente.');
        repeat=true;
    end
end

repeat=true;
while repeat
    time_hour = input ('Insira a hora [HH:MM:SS]: \n', 's');
    if length(time_hour)==8
        if strcmp(time_hour(3),':')&& strcmp(time_hour(6),':')&& ~
            isnan(str2double(time_hour(1:2)))&& ~isnan(str2double(
            time_hour(4:5)))&& ~isnan(str2double(time_hour(7:8)))
            repeat=false;
            hour=str2double(time_hour(1:2));
            min=str2double(time_hour(4:5));
            sec=str2double(time_hour(7:8));
        else
            disp('Insira novamente.');
            repeat=true;
        end
    else
        disp('Insira novamente.');
        repeat=true;
    end
end

time = 24*60*60*datenum(year,month,day,hour,min,sec); % Number of
    seconds relative to a certain date
```

```
function data=compute_orbit_parameters(data,ts0,tsim)
% List of variables for YUMA ALMANACS:
%% id     = ID
%% e      = Eccentricity
%% TOA    = Time of applicability
%% alpha  = Orbital inclination
%% asc_r  = Rate of right ascension
%% sqrt_A = Square root of semi-major axis
%% ASC    = Right ascencion at time of Almanac
```

```
% % w         = Argument of Perigee
% % M         = Mean anomaly
% % afo
% % af1
% % week

t_st=ts0+tsim; % Transmission time is equal to the receiver's
    current time plus the trajectory simulation time
t_oe= 24*60*60*datenum(2019,01,13,12,12,12); % the seconds of when
    the almanac was issued
delta_t = t_st - t_oe;
const_grav=3.986005e14;
for i=1:length(data)
    % Compute the mean motion n
    n(i)=sqrt(const_grav/((data(i).sqrt_A)^6));

    % Compute Mean anomaly
    data(i).M=data(i).M0 + n(i)*delta_t;

    % compute Eccentric anomaly (E) (iterative mode)
    E_int(1)= data(i).M + (((data(i).e)*sin((data(i).M)))/(1-sin((
        data(i).M)+(data(i).e))+sin((data(i).M))));
    for j=2:20
        E_int(j)= data(i).M + data(i).e*sin(E_int(j-1));
    end
    data(i).E=E_int(20);

    % compute true anomaly (true_anm)
    data(i).true_anm = atan2 (( sqrt(1-(data(i).e)^2)*sin(data(i).E)
        ),(cos(data(i).E)-data(i).e));

    % theta: argument of latitude
    data(i).theta = data(i).true_anm + (data(i).w);

    %------------------------------------------------------------------%
    % Longitude of the ascending node (omega)
    omega0=data(i).ASC;
    data(i).omega=omega0+(data(i).asc_r)*delta_t-(2*pi*t_st)/86164;


    %------------------------------------------------------------------%
    % Computation of orbit radius(R): distance between the satellite
        and the
    % Earth center
    data(i).R = ((data(i).sqrt_A)^2)*(1-(data(i).e)*cos((data(i).E))
        );

end
end
```

```
function data=compute_ECEF_coordinates(data)
```

```

% ECEF coordinates of each GPS of the Almanac:
for i=1:length(data)
    data(i).x =data(i).R*(cos(data(i).theta)*cos(data(i).omega)-sin(
       data(i).theta)*sin(data(i).omega)*cos(data(i).alpha));
    data(i).y =data(i).R*(cos(data(i).theta)*sin(data(i).omega)+sin(
       data(i).theta)*cos(data(i).omega)*cos(data(i).alpha));
    data(i).z =data(i).R*sin(data(i).theta)*sin(data(i).alpha);
end
```

```
function [receiver_coord_lla] = requests_coordinates_user()
% This function requests receiver's coordinates to the user.

fprintf('Insira as coordenadas do recetor:\n');
repeat=true;
coord(1,3)=0;
while repeat
    lat_receiver=input('Insira a latitude do recetor: DD-MM-SS N/S \
       n', 's');
    [repeat, coord(1,1)]=test_coordinates_user(repeat, lat_receiver,
        '1');
end

repeat=true;
while repeat
    lon_receiver=input('Insira a longitude do recetor: DD-MM-SS W/E
       \n', 's');
    [repeat, coord(1,2)]=test_coordinates_user(repeat, lon_receiver,
        2);
end

repeat=true;
while repeat
    alt_receiver=input('Insira a altitude do recetor: [m] ', 's');
    [repeat, coord(1,3)]=test_coordinates_user(repeat, alt_receiver,
        3);
end
receiver_coord_lla = coord;
clc;
disp('Coordenadas do recetor inseridas:')
fprintf('Latitude:  %.3f \n',coord(1,1))
fprintf('Longitude: %.3f \n',coord(1,2))
fprintf('Altitude:  %.3f m\n',coord(1,3))
```

```
function mask_angle=requests_mask_angle
% Requests mask angle
repeat=true;
while repeat
    mask_input = input ('\nInsira o  gulo  de m scara [ ] \n', 's');
    mask_angle = str2double(mask_input);
    if ~isnan(mask_angle)
```

```
            repeat=false;
        end
end
```

```
function [NView,visible_sat, data] = satellites_in_view(receiver_coord, data, mask_angle)
% In order to compute satellites in view it is necessary:
%%% Mask angle
%%% Satellits ENU coordinates (from ECEF satellits coordinates and geodetic receiver's coordinates)
%%% Azimute of satellites
%%% Elevation of satellites

%% Convert satellite ECEF coordinates to satellite ENU coordinates
%% Computes salellite azimute and elevation
%% Compares satellite elevation angle with mask angle:
%%%% if elevation >= mask angle : OK
j=1;
visible_sat(j).id = 0;
visible_sat(j).el = 0;
visible_sat(j).az = 0;
visible_sat(j).x  = 0;
visible_sat(j).y  = 0;
visible_sat(j).z  = 0;

for i=1:length(data)
    [data(i).ENU_E,data(i).ENU_N,data(i).ENU_U]=ecef2enu(data(i).x, data(i).y, data(i).z, receiver_coord(1), receiver_coord(2), receiver_coord(3), wgs84Ellipsoid);
    [data(i).AZ, data(i).EL]=compute_azim_elevation(data(i).ENU_E, data(i).ENU_N,data(i).ENU_U);

    if data(i).EL>=mask_angle
        visible_sat(j).id=data(i).id;
        visible_sat(j).el=data(i).EL;
        visible_sat(j).az=data(i).AZ;
        visible_sat(j).x = data(i).x;
        visible_sat(j).y = data(i).y;
        visible_sat(j).z = data(i).z;
        j=j+1;
    end
end

if isnan(mask_angle) % Canyon scenario: choose the four satellites with biggest elevation angle
    for i=1:length(data)
        B(i)=data(i).EL;
    end
    B_orden = sort(B,'descend');
    B_big= B_orden(1:4);
    for j=1:length(B_big)
```

```
            for i=1:length(data)
                if B_big(j)==data(i).EL
                    visible_sat(j).id=data(i).id;
                    visible_sat(j).el=data(i).EL;
                    visible_sat(j).az=data(i).AZ;
                    visible_sat(j).x = data(i).x;
                    visible_sat(j).y = data(i).y;
                    visible_sat(j).z = data(i).z;
                end
            end
        end
end

NView=length(visible_sat);
end
```

```
function N=requests_nr_sattelite(NView)

repeat=true;
disp('Selecione o n mero de sat lites que deseja utilizar neste percurso: ');
while repeat
    N=input('','s');
    N=str2double(N);
     if isnan(N)|| N>NView|| N<4
        repeat=true;
        fprintf('O n mero inserido deve ser inteiro, superior a 4 sat lites e menor que %d sat lites. \nTente novamente.\n', NView);
     else
        repeat=false;
    end
end
```

```
function [minPDOP_sat_const, pdopmin ,total_comb,PDOP_vector]= PDOP_min(visible_sat, ECEF_coord_r, NView, N, path_ecef)
% Determination of the optimum sub-constellation for a certain number of satellites N selectable
% (with NView >= N >=4) using minimization of the PDOP parameter.
% % % Requests number N of satellits

if ~isnan(path_ecef) % IF IS NOT CANYON SCENARIO

    % Number of possible combinations
    nr_comb = nchoosek(NView,N);

    % Shows every combination of satellites index's possible in NView space
    % Example: Nview=3 ; N=2; computes: [1 2];[1 3]; [2 3]
    total_comb = nchoosek((1:NView),N);

```

```
    % Computes PDOP vector
    PDOP_vector = PDOP(visible_sat, ECEF_coord_r, N, total_comb, nr_comb);

    % Finds PDOP min
    [ pdopmin , position] = min(PDOP_vector);

    for i=1:N
        minPDOP_sat_const(i) = visible_sat(total_comb(position,i));
    end

else % IF IT IS CANYON SCENARIO
     total_comb = [1 2 3 4];
     PDOP_vector = PDOP(visible_sat, ECEF_coord_r, 4, total_comb, 1);
     [ pdopmin , position] = min(PDOP_vector);
     minPDOP_sat_const = visible_sat;
end

end
```

```
function [PDOP_vector,G]= PDOP(visible_sat, ECEF_coord_r, N, total_comb, nr_comb)
% This function PDOP algorithm

% Computes every distance between satellite and receiver
for i=1:length(visible_sat)
    dist(i)=sqrt((visible_sat(i).x-ECEF_coord_r(1))^2 + (visible_sat(i).y-ECEF_coord_r(2))^2+ (visible_sat(i).z-ECEF_coord_r(3))^2);
end

% Computes G matrix (N X 4)X N DE COMBINAES POSSVEIS DE N EM NVIEW
for j=1:nr_comb
    for i=1:N
        G(i,1,j)= -(visible_sat(total_comb(j,i)).x- ECEF_coord_r(1))/dist(total_comb(j,i));% 1st column
        G(i,2,j)= -(visible_sat(total_comb(j,i)).y- ECEF_coord_r(2))/dist(total_comb(j,i));% 2nd column
        G(i,3,j)= -(visible_sat(total_comb(j,i)).z- ECEF_coord_r(3))/dist(total_comb(j,i));% 3rd column
        G(i,4,j)=  1;
    end
    % Compute PDOP for each (N X 4) matrix
    H(:,:,j)=inv((G(:,:,j).')*G(:,:,j));
    PDOP_vector(j)=sqrt(H(1,1,j)+H(2,2,j)+H(3,3,j));

end

end
```

```
function creates_polar_graphs(visible_sat,minPDOP_sat_const, data)
for i=1:length(visible_sat)
    azv(i)=visible_sat(i).az;
    elv(i)=visible_sat(i).el;
end

for i=1:length(minPDOP_sat_const)
    az(i)=minPDOP_sat_const(i).az;
    el(i)=minPDOP_sat_const(i).el;
end

for i=1:length(data)
    azgps(i)=data(i).AZ;
    elgps(i)=data(i).EL;
end

figure();
polar(azv,elv,'o')
hold on
polar(az,el,'*')
hold on
polar (0,0,'+')
legend('Sat lites vis veis totais','Sub-constela   ma ','Receptor
    ')

figure();
polar(azv,elv,'o')
legend('Sat lites v siveis totais');

figure();
polar(azgps, elgps,'o');
legend('Conjunto total de sat lites GPS em  ita ')
```

```
function path=trajectory()

%% AB
% time interval = 100 seconds;
% P: Real ecef position init
x_0=0;
y_0=0;
a=1;
v_0=50;

t=[0:100];
for i=1:101
    y(i)=y_0+v_0*t(i)+0.5*a*t(i).^2;
    v(i)=v_0+a*t(i);
end
x(1:101)=x_0;

%% BC
```

```
% time interval = 50
t=[1:50];
w=(3*pi/2)/50;
R=v(101)/w;
centro(1)=x(101)-R;
centro(2)=y(101);
for i=1:50
    theta(i)=w*t(i); %rad
    x(101+i)=centro(1)+R*cos(theta(i));
    y(101+i)=centro(2)+R*sin(theta(i));
    v(101+i)=v(101);
end


% % CD
% % time interval = 50;
for i=1:50
    x(151+i)= x(151)+v(101)*t(i);
    y(151+i)= y(151);
    v(151+i)= v(101);
end


% % DE : canyon scenario
% % time interval = 50 ;
for i=1:50
    x(201+i)= x(201)+v(101)*t(i);
    y(201+i)= y(201);
    v(201+i)= v(101);
end

path(:,1)= x ;
path(:,2)= y ;
path(:,3)= 0 ;          % z = 0 because its a 2D moviment
path((1:201),5)=10;     % Mask angle 10
path((202:end),5)=10; % Canyon scenario
path(:,6) = v;
```

```
function option_menus=menu()

repeat=true;
fprintf('\nEscolha um dos seguintes m todos para estimar a sua posi  ao longo do percurso: \n');
fprintf('1) M todo dos m nimos quadrados\n2) Filtro de Kalman (Modelo PV)\n');
fprintf('3) Sair do programa.\n');
while repeat
    option_menus=input('','s');
    if strcmp(option_menus,'1') || strcmp(option_menus,'2')|| strcmp(option_menus,'3')
        repeat=false;
```

```
    else
        disp('Op  invlida. Tente novamente.')
    end
end
end
```

```
function variance=request_variance
% Requests variance [m^2]
repeat=true;
while repeat
    variance=str2double(input('Insira a varincia pretendida: [m^2] \n','s'));
    if isnan(variance) || variance<0
        disp('Insira um nmero. Tente novamente.');
    else
        repeat = false;
    end
end
```

## 6.3 Código dos Mínimos Quadrados

```
function [position_estimated]= least_square_algorithm(
    minPDOP_sat_const, variance, position, position_est)

%%% Pseudorange computation algorithm
%%%% Requests variance
%%%% Computes noise (AWGN) with selected variance
%%%% Computes ionosphere/troposphere error
%%%% Computes pseudo_ranges_meas vector
%%% Least-Squares algorithm
%%%% Given a initial position (estimated) we obtain pseudo_range
    estimated
%%%% Given pseudo_range measured ( with noite + I)


for j=1:length(minPDOP_sat_const)
        dist(j) =sqrt((minPDOP_sat_const(j).x-position(1))^2 + (
            minPDOP_sat_const(j).y-position(2))^2+ (minPDOP_sat_const
            (j).z-position(3))^2) + position(4) ;
        total_noise = noise(minPDOP_sat_const(j).el,variance, 1);
                            % Ionospheric error option on
        pseudo_range_meas(j) = dist(j) + total_noise;
                                            % TOTAL NOISE = noise + I
end

for j=1:length(minPDOP_sat_const)
        dist_est(j) =sqrt((minPDOP_sat_const(j).x-position_est(1))^2
             + (minPDOP_sat_const(j).y-position_est(2))^2+ (
            minPDOP_sat_const(j).z-position_est(3))^2);
        pseudo_range_estim(j)= dist_est(j) + position_est(4);
end

for k=1:length(minPDOP_sat_const)
    G(k,1)= -(minPDOP_sat_const(k).x- position_est(1))/dist_est(k);
        % 1st column
    G(k,2)= -(minPDOP_sat_const(k).y- position_est(2))/dist_est(k);
        % 2nd column
    G(k,3)= -(minPDOP_sat_const(k).z- position_est(3))/dist_est(k);
        % 3rd column
    G(k,4)= 1;
end

for k=1:length(minPDOP_sat_const)
    delta_pseudo(k) = pseudo_range_meas(k) - pseudo_range_estim(k);
end

delta_x=(inv(transpose(G)*G)*transpose(G))*transpose(delta_pseudo);
delta_x=delta_x.';

```

```
position_estimated = position_est+ delta_x; % Repeatedly corrects the position error


end
```

## 6.4 Código do Filtro de Kalman Linearizado

```
function [position_est, Xest_prev, phi, Pprev]=kalman_filter(
    minPDOP_sat_const,variance,option, path, Xest_prev, phi, Pprev)


%% Kalman dynamics (PV model)
% Initial conditions
I=eye(8);

c=3*10^8;

%%% Observation noise error covariance matrix : R
R=eye(length(minPDOP_sat_const))*variance;

%% Noise covariance matrix with receiver's clock as a
% quartz temperature compensated oscillator. (2 state model)
h0=2*10^(-19);
h2=2*10^(-20);
qfase=h0/2*c^2;
qf=2*pi^2*h2*c^2;
qv=10;
Q=zeros(8);
Q(1,1)=qv/3;
Q(3,3)=qv/3;
Q(5,5)=qv/3;
Q(1,2)=qv/2;
Q(2,1)=qv/2;
Q(3,4)=qv/2;
Q(4,3)=qv/2;
Q(5,6)=qv/2;
Q(6,5)=qv/2;
Q(2,2)=qv;
Q(4,4)=qv;
Q(6,6)=qv;
Q(7,7)=qfase+qf/3;
Q(7,8)=qf/2;
Q(8,7)=qf/2;
Q(8,8)=qf;


xu=path(:,1);
yu=path(:,2);
zu=path(:,3);
tu=0;


%% Compute Observation matrix
for n=1:length(minPDOP_sat_const)
    dist_est(n)= sqrt((minPDOP_sat_const(n).x-Xest_prev(1))^2+(
```

```
        minPDOP_sat_const(n).y-Xest_prev(3))^2+(minPDOP_sat_const(n).
        z-Xest_prev(5))^2);
    h(n,1)= dist_est(n)+ Xest_prev(7); % adds c*time drift estimated
         component % Observation vector estimated
    H(n,1)= - (minPDOP_sat_const(n).x-Xest_prev(1))/dist_est(n);
    H(n,3)= - (minPDOP_sat_const(n).y-Xest_prev(3))/dist_est(n);
    H(n,5)= - (minPDOP_sat_const(n).z-Xest_prev(5))/dist_est(n);
    H(n,7)= 1;
    H(n,8)= 0;
end



%%  Observation vector Z measured = pseudoranges
for n=1:length(minPDOP_sat_const)
    v(n,1)=noise(minPDOP_sat_const(n).el,variance,option); % Total
        noise (v) = gaussian + iono
    Z(n,1)=sqrt((minPDOP_sat_const(n).x-xu)^2+(minPDOP_sat_const(n).
        y-yu)^2+(minPDOP_sat_const(n).z-zu)^2)+ Xest_prev(7) + v(n,1)
        ; % xu, vu, zu, tu initial true position + total noise
end

%% FILTERING STEP

%%% Kalman gain
K=Pprev*transpose(H)*inv(H*Pprev*transpose(H)+R);

%%% Estimate update
Xest=Xest_prev+K*(Z-h);

%%% Error covariance update
P=(I-K*H)*Pprev*transpose(I-K*H)+K*R*transpose(K);

%% PREDICTION STEP
Xest_prev=phi*Xest;
Pprev=phi*P*phi.'+Q;
position_est = transpose(Xest);
end
```